New Challenges in Detection and Management of Security Vulnerabilities in Data Networks
Tesi Doctoral

Doctorand	:	Guiomar Corral Torruella
Directora del Treball	:	Dra. Elisabet Golobardes i Ribé
Programa de Doctorat	:	Tecnologies de la Informació i les Comunicacions i la seva gestió

Grup de Recerca en Sistemes Intel·ligents
Escola Tècnica Superior d'Enginyeria en Electrònica i Informàtica
Enginyeria i Arquitectura La Salle – Universitat Ramon Llull
Quatre Camins, 2 – 08022 Barcelona
http://www.salle.url.edu/GRSI

10 de 7 de 2009

Abstract

As networks become an integral part of corporations and everyone's lives, advanced network security technologies are being developed to protect data and preserve privacy. Network security testing is necessary to identify and report vulnerabilities, and also to assure enterprise security requirements. Security analysis is necessary to recognize malicious data, unauthorized traffic, detected vulnerabilities, intrusion data patterns, and also to extract conclusions from the information gathered in the security test. Then, where is the problem? There is no open-source standard for security testing, there is no integral framework that follows an open-source methodology for security testing, information gathered after a security test includes large data sets, there is not an exact and objective pattern of behavior among network devices or, furthermore, among data networks and, finally, there are too many potentially vulnerabilities. The challenge of this domain resides in having a great volume of data; data are complex and can appear inconsistent diagnostics. It is also an unsupervised domain where no machine learning techniques have been applied before. Thus a complete characterization of the domain is needed.

Consensus is the main contribution of this thesis. *Consensus* is an integrated framework that includes a computer-aided system developed to help security experts during network testing and analysis. The system automates mechanisms related to a security assessment in order to minimize the time needed to perform an OSSTMM security test. This framework can be used in wired and wireless networks. Network security can be evaluated from inside or from outside the system. It gathers data of different network devices, not only computers but also routers, firewalls and Intrusion Detection Systems (IDS). *Consensus* manages many data to be processed by security analysts after an exhaustive test. General information, port scanning data, operating system fingerprinting , vulnerability scanning data, routing and filtering rules, IDS response, answer to malicious code, weak passwords reporting, and response to denial of service attacks can be stored for each tested device. This data is gathered by the automated testing tools that have been included in *Consensus*.

The great amount of data for every device and the different number and type of attributes complicates a manually traffic pattern finding. The automated testing tools can obtain different results, incomplete or inconsistent information. Then data obtained from a security test can be uncertain, approximate, complex and partial true. In this environment arises the second main contribution of this thesis: *Analia*, the data analysis module of *Consensus*. Whereas *Consensus* gathers security data, *Analia* includes Artificial Intelligence to help analysts after a vulnerability assessment. Unsupervised learning has been analyzed to be adapted to this domain. *Analia* finds resemblances within tested devices and clustering aids analysts in the extraction of conclusions. Afterwards, the best results are selected by applying cluster validity indices. Then explanations of clustering results are included to give a more comprehensive response to security analysts.

The combination of machine learning techniques in the network security domain provides benefits and improvements when performing security assessments with the *Consensus* framework and processing its results with *Analia*.

Resumen

A medida que las redes pasan a ser un elemento integral de las corporaciones, las tecnologías de seguridad de red se desarrollan para proteger datos y preservar la privacidad. El test de seguridad en una red permite identificar vulnerabilidades y asegurar los requisitos de seguridad de cualquier empresa. El análisis de la seguridad permite reconocer información maliciosa, tráfico no autorizado, vulnerabilidades de dispositivos o de la red, patrones de intrusión, y extraer conclusiones de la información recopilada en el test. Entonces, ¿dónde está el problema? No existe un estándar de código abierto ni un marco integral que siga una metodología de código abierto para tests de seguridad, la información recopilada después de un test incluye muchos datos, no existe un patrón exacto y objetivo sobre el comportamiento de los dispositivos de red ni sobre las redes y, finalmente, el número de vulnerabilidades potenciales es muy extenso. El desafío de este dominio reside en tener un gran volumen de datos complejos, donde pueden aparecer diagnósticos inconsistentes. Además, es un dominio no supervisado donde no se han aplicado técnicas de aprendizaje automático anteriormente. Por ello es necesaria una completa caracterización del dominio.

Consensus es la aportación principal de esta tesis: un marco integrado que incluye un sistema automatizado para mejorar la realización de tests en una red y el análisis de la información recogida. El sistema automatiza los mecanismos asociados a un test de seguridad y minimiza la duración de dicho test, siguiendo la metodología OSSTMM. Puede ser usado en redes cableadas e inalámbricas. La seguridad se puede evaluar desde una perspectiva interna, o bien externa a la propia red. Se recopilan datos de ordenadores, *routers*, *firewalls* y detectores de intrusiones. *Consensus* gestionará los datos a procesar por analistas de seguridad. Información general y específica sobre sus servicios, sistema operativo, la detección de vulnerabilidades, reglas de encaminamiento y de filtrado, la respuesta de los detectores de intrusiones, la debilidad de las contraseñas, y la respuesta a código malicioso o a ataques de denegación de servicio son un ejemplo de los datos a almacenar por cada dispositivo. Estos datos son recopilados por las herramientas de test incluidas en *Consensus*.

La gran cantidad de datos por cada dispositivo y el diferente número y tipo de atributos que les caracterizan, complican la extracción manual de un patrón de comportamiento. Las herramientas de test automatizadas pueden obtener diferentes resultados sobre el mismo dispositivo y la información recopilada puede llegar a ser incompleta o inconsistente. En este entorno surge la segunda principal aportación de esta tesis: *Analia*, el módulo de análisis de *Consensus*. Mientras que *Consensus* se encarga de recopilar datos sobre la seguridad de los dispositivos, *Analia* incluye técnicas de Inteligencia Artificial para ayudar a los analistas después de un test de seguridad. Distintos métodos de aprendizaje no supervisado se han analizado para ser adaptados a este dominio. *Analia* encuentra semejanzas dentro de los dispositivos analizados y la agrupación de dichos dispositivos ayuda a los analistas en la extracción de conclusiones. Las mejores agrupaciones son seleccionadas mediante la aplicación de índices de validación. A continuación, el sistema genera explicaciones sobre cada agrupación para dar una respuesta más detallada a los analistas de seguridad.

La combinación de técnicas de aprendizaje automático en el dominio de la seguridad de redes proporciona beneficios y mejoras en la realización de tests de seguridad mediante la utilización del marco integrado *Consensus* y su sistema de análisis de resultados *Analia*.

Resum

A mesura que les xarxes passen a ser un element integral de les corporacions, les tecnologies de seguretat de xarxa es desenvolupen per protegir dades i preservar la privacitat. El test de seguretat en una xarxa permet identificar vulnerabilitats i assegurar els requisits de seguretat de qualsevol empresa. L'anàlisi de la seguretat permet reconèixer informació maliciosa, tràfic no autoritzat, vulnerabilitats de dispositius o de la xarxa, patrons d'intrusió, i extreure conclusions de la informació recopilada en el test. Llavors, on està el problema? No existeix un estàndard de codi obert ni un marc integral que segueixi una metodologia de codi obert per a tests de seguretat, la informació recopilada després d'un test inclou moltes dades, no existeix un patró exacte i objectiu sobre el comportament dels dispositius de xarxa ni sobre les xarxes i, finalment, el nombre de vulnerabilitats potencials és molt extens. El desafiament d'aquest domini resideix a tenir un gran volum de dades complexes, on poden aparèixer diagnòstics inconsistents. A més, és un domini no supervisat on no s'han aplicat tècniques d'aprenentatge automàtic anteriorment. Per això cal una completa caracterització del domini.

Consensus és l'aportació principal d'aquesta tesi: un marc integrat que inclou un sistema automatitzat per millorar la realització de tests en una xarxa i l'anàlisi de la informació recollida. El sistema automatitza els mecanismes associats a un test de seguretat i minimitza la durada de l'esmentat test, seguint la metodologia OSSTMM. Pot ser usat en xarxes cablejades i sense fils. La seguretat es pot avaluar des d'una perspectiva interna, o bé externa a la pròpia xarxa. Es recopilen dades d'ordinadors, *routers*, *firewalls* i detectors d'intrusions. *Consensus* gestionarà les dades a processar per analistes de seguretat. Informació general i específica sobre els seus serveis, sistema operatiu, la detecció de vulnerabilitats, regles d'encaminament i de filtrat, la resposta dels detectors d'intrusions, la debilitat de les contrasenyes, i la resposta a codi maliciós o a atacs de denegació de servei són un exemple de les dades a emmagatzemar per cada dispositiu. Aquestes dades són recopilades per les eines de test incloses a *Consensus*.

La gran quantitat de dades per cada dispositiu i el diferent número i tipus d'atributs que els caracteritzen, compliquen l'extracció manual d'un patró de comportament. Les eines de test automatitzades poden obtenir diferents resultats sobre el mateix dispositiu i la informació recopilada pot arribar a ser incompleta o inconsistent. En aquest entorn sorgeix la segona principal aportació d'aquesta tesi: *Analia*, el mòdul d'anàlisi de *Consensus*. Mentre que *Consensus* s'encarrega de recopilar dades sobre la seguretat dels dispositius, *Analia* inclou tècniques d'Intel·ligència Artificial per ajudar als analistes després d'un test de seguretat. Diferents mètodes d 'aprenentatge no supervisat s'han analitzat per ser adaptats a aquest domini. *Analia* troba semblances dins dels dispositius analitzats i l'agrupació dels esmentats dispositius ajuda als analistes en l'extracció de conclusions. Les millors agrupacions són seleccionades mitjançant l'aplicació d'índexs de validació. A continuació, el sistema genera explicacions sobre cada agrupació per donar una resposta més detallada als analistes de seguretat.

La combinació de tècniques d'aprenentatge automàtic en el domini de la seguretat de xarxes proporciona beneficis i millores en la realització de tests de seguretat mitjançant la utilització del marc integrat *Consensus* i el seu sistema d'anàlisi de resultats *Analia*.

Acknowledgments

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. All this work has been possible with the support of the *Grup de Recerca en Sistemes Intel·ligents* of La Salle (URL) and specially, the members that helped me in the different publications: Albert Fornells, Álvaro Garcia, Albert Orriols and David Vernet. I would also like to thank Eva Armengol, from the IIIA (Artificial Intelligence Research Institute), for her collaboration in this thesis. The discussions and cooperations with all of them have contributed substantially to this work.

I am thankful to my supervisor, Dr. Elisabet Golobardes, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject, approaching two initial separate worlds: the telematics and the artificial intelligence.

I am grateful to the Telematics team as well: Agustín, Àlex, Josep Maria, and specially, Jaume. It is a pleasure to thank those who made Consensus and Analia possible from the first beginning: Gaspar, Jordi, Maite, Xavi, Enrique, Albert, Angel, Anna, Elisabet, Àngel, Hector, Omar, Esther, Alex, Isard and Marc. I am also indebted to Pete Herzog for all his help, support, interest and valuable hints, specially with the OSSTMM. Besides, I want to thank Eva for her assistance with refining the English written.

I would also like to thank the *Ministerio de Educación y Ciencia*, which has supported the MID-CBR (TIN2006-15140-C03-03) and VDS-Analia (CIT-390000-2005-27) projects, and the *Ministerio de Industria, Turismo y Comercio*, which has supported the Consensus-VDS project (FIT-360000-2004-81). I want to thank Enginyeria La Salle for its support during this long way.

I cannot end without thanking my family, on whose constant encouragement and unconditional support I have relied throughout my time spent in this thesis. Moltes gràcies Gustau, moltes gràcies pares, per estar allí des del principi. I moltes gràcies Biel, el més petit de la família, però que m'ha donat la darrera empenta per presentar aquesta tesi.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Framework and Related Work

Chapter 1

Introduction

The significant need of security systems that integrate malware and vulnerability detection in large-scale operational communication networks has become the leitmotiv of this thesis. One of the main goals has been the automation of the processes related to a security assessment following a well-known and established open-source methodology for security testing. The second main goal has been focused on the approaching between the security field and the artificial intelligence in order to get benefit from the best of both worlds. The analysis of data gathered in a security test has been improved with the application of artificial intelligence techniques. This chapter introduces the main concepts of this thesis, describes the framework, and details the motivation and objectives. It concludes with a formal overview of the organization of this document.

1.1 Introduction

The growing requirement of the network market for functional intelligent systems in the security area is currently motivating important research and development work. Intelligent systems, which can provide human like expertise such as domain knowledge and uncertain reasoning, are important in tackling practical computing problems. This thesis aims to provide a contribution to the network security field including the benefits of intelligent systems. This intrinsic interconnection will benefit from the best of both worlds. Its result will be a new framework for security assessments that improves the security testing phase and the analysis of testing results. The complete framework has been named *Consensus* and the intelligent system module has been called *Analia*. The integration of different learning techniques, to overcome individual limitations and achieve synergetic effects through combination of these techniques, will contribute to a more complete intelligent system for security testing purposes. The experimentation will be presented to demonstrate how this novel framework has been used to solve real world problems.

This thesis comes up to new challenges in detection and management of security vulnerabilities in data networks. It researches a new assessment framework for the testing of communication networks in terms of security. First of all, the different processes that have to be carried out in a security test according to a particular methodology are investigated. The most appropriate processes are selected to become automated in order to improve the testing phase of a security assessment. This main contribution materializes in the framework *Consensus*. It provides a new model to perform security tests by automating the processes with the incorporation of open-source security tools in the system. The management of the information collected after a security test is also investigated in this thesis. Machine learning techniques are analyzed and the most suitable ones are selected to be integrated in the thesis framework. The combination of these machine learning approaches gets the best of them and overcomes their individual deficiencies to produce the second main contribution of this thesis, *Analia*, the analysis module of *Consensus* with Artificial Intelligence (AI) capabilities.

This chapter presents a comprehensive overview of the thesis. The framework that embraces this thesis is detailed and the lines of research are related to the different projects in which we have taken part. The fundamentals that motivated this research and the reasons to contribute to the research community with a new security assessment framework are detailed. The main goals of this thesis are also described in this chapter. Finally, the organization of this thesis is illustrated.

1.2 Framework

This thesis has been submitted to the PhD program *Tecnologies de la Informació i les Comunicacions i la seva gestió* (Doctorate in Information and Communication Technologies and its Management) of the Universitat Ramon Llull (URL). This thesis is part of the research of the *Grup de Recerca en Sistemes Intel·ligents* (Research Group in Intelligent Systems), GRSI, of Enginyeria La Salle of the URL.

The research of the GRSI is focused on Machine Learning, especially on the field of Knowledge Discovery from Databases, also known as Data Mining. The research group aims at extracting interesting patterns from moderate and large data sets. In this framework, GRSI works on different stages of the process of data mining: pre-processing, characterization of data sets, analysis for a better understanding and improvement of machine learning techniques, methodologies to evaluate learners and post-processing. Over the last few years, the research has mainly focused on learning methods inspired in natural principles and analogy. The group is known for its expertise on Evolutionary Computation, Case-Based Reasoning and Soft Computing, and Data Complexity. The group seeks for applying its research to real world domains which may benefit society in the near future. Some of its recent applications are the support to the diagnosis of breast cancer and the computer-aided analysis of attacks in computer networks. Currently, the group is also working on other challenging medical and industry applications. The group has been accredited by *Generalitat de Catalunya* as Consolidated Research Group (2002 SGR-00155, 2005 SGR 302).

One of the domains of application in which GRSI researches into is the security on data networks. This thesis forms part of this domain and presents an organized approach where machine learning and intelligent systems become valid alternatives to solve existent problems in the data network domain. All these problems have been arisen in the different projects where the work and the different contributions of this thesis have been included.

I joined the GRSI in 1998, more than ten years ago. My collaboration with the GRSI started with the project Design and Implementation of a New Generation Telematics Platform to Support Open and Distance Learning (*Implementación y estudio de herramientas de inteligencia artificial aplicadas a plataformas de enseñanza abierta a distancia y a las redes ATM que las soportan*). This study was partially supported by the Spanish Education Ministry grant CICYT-TEL-98-0408. In this project the cooperation between the artificial intelligence and the data network fields started from the beginning, proving that artificial intelligence techniques can provide useful solutions to network problems.

The research work in the security domain started in the project Consensus - Vulnerability Detection System, partially funded by the *Ministerio de Industria, Turismo y Comercio* under grant number FIT-360000-2004-81. This work continued with the project *VDS-ANALIA: Sistema de análisis de detección de vulnerabilidades mediante inteligencia artificial* (VDS-ANALIA: Vulnerability Detection Analysis System by means of Artificial Intelligence) funded by the *Ministerio de Educación y Ciencia* under grant CIT-390000-2005-27. The line of research was also integrated in the CICYT project called MID-CBR: A Unified Framework for the Development of Case-Based Reasoning (CBR) Systems (*Un Marco Integrador para el Desarrollo de Sistemas de Razonamiento Basado en Casos*). This project has been supported by the Spanish Ministry of Education and Science under grant TIN2006-15140-C03-03.

A brief description of these projects is detailed below:

- **IA-ATM** (CICYT-TEL-98-0408): This project had two main goals for the GRSI: the application of AI to open and distance learning environments, and also to ATM networks. The first goal pursued to predict the evolution of the students to design specific plans to avoid future fails. The second goal was related to the prediction of congestion in a ATM network to take measures in advance so as not to get the network congested [32]. Partners: Universitat de Girona and Universitat Oberta de Catalunya.

- **Consensus** (FIT-360000-2004-81): The main goal of this project was to develop a distributed tool that was dedicated to security environment professionals. The tool follows the Open Source Security Testing Methodology Manual (OSSTMM) to carry out vulnerability tests in an autonomous and distributed way. Partner: ISECOM, who created the OSSTMM.

- **Analia** (CIT-390000-2005-27): The objective of this project was the application of Artificial Intelligence techniques to the improvement of the analysis of the information obtained while carrying out security vulnerability tests in data networks. Partner: ISECOM.

- **MID-CBR** (TIN2006-15140-C03-03): The main objectives of this project related to this thesis are the study of new ways to use soft computing techniques for CBR, techniques for case retrieval in knowledge-intensive CBR systems and the empirical evaluation of the developed techniques by means of CBR prototypes implemented for several experimental domains. In this thesis the experimental domain was the data network security. Partners: Universidad Complutense de Madrid and the Artificial Intelligence Research Institute. Entities of promotion and observation: ISECOM and the *Fundació Clínic per a la Recerca Biomèdica*.

In parallel, I have collaborated on other research projects related to the networking area but without the component of Artificial Intelligence. These projects focus mainly on security aspects of different technologies of data networks. All these projects have been carried out in La Salle as well, led by the Telematics group. An abstract of these projects is detailed below:

- **ATM-VP**: The project Adaptative Reconfiguration of ATM virtual networks based on Virtual Paths was done together with the corporation DIMAT in 1998. The main goal of this project was to develop a design and an implementation of an automatic management console of the ATM virtual networks in case of congestion [92].

- **CSI-RHET**: The project Quality of Service (QoS) and Traffic Engineering in Heterogeneous Network addressed the design, implementation and evaluation of a heterogeneous multiservice broadband telecommunications network based on IPv6 with QoS. Partners: Universidad de Cartagena, Instituto Tecnológico de Aragón, Universidad de Alcalá de Henares.

- **OPERA 2**: The main goal of the project Open PLC European Research Alliance Phase 2 was to develop a Power Line Communications (PLC) technology that can become an alternative for a broadband access integrated network. The URL goal was to perform a security analysis of the PLC technology to evaluate the OPERA1 specification and contribute with a new proposal for the standardization process (2005-2008).

- **ICARO**: It stands for Integrity, Confidentiality, Authenticity and secuRity in pOwerline networks. This project researches about the advanced security mechanisms adapted to PLC in terms of data integrity, confidentiality and authenticity. This project has been done together with the company DS2 from 2008 until 2009. The conclusions extracted from this project are being used to the ITU-T standardization for PLC. This project has been partially funded by the *Ministerio de Industria, Turismo y Comercio* in the Avanza framework.

1.3 Motivation

The security of communication and data networks was not considered a priority a few years ago, perhaps because nobody was able to predict such an impressive growth of the use of data networks and their significant importance worldwide. The design of protocols, devices and networks was more focused on their operational function rather than providing systems that fulfilled security requirements. However this trend has radically changed by now. Nowadays this is a very prolific line of research with many efforts dedicated to security, as it can be extracted from the european Framework Programme (FP). As an example, the VII FP 2007-2013 has planned to invest 1,400 million euros in the security theme in 7 years, 4% of the total budget[1]. One of the cross cutting activities included in the research themes is the security systems integration, interconnectivity and interoperability, where the work of this thesis adapts to the topics proposed.

The need of improvement of the security testing mechanisms in data networks in order to enrich the threat and vulnerability detection process is a red-hot issue. Hackers tend to go a step forward security systems to break barriers and enter into public networks and systems. So intelligent systems are indispensable to discover the own security flaws with the purpose of overtaking possible hacking attacks. One of the main contributions of this thesis formulates a solution for this matter. A new system that automates the processes related to a security test following an open-source testing methodology widely disseminated, like the OSSTMM (Open Source Security Testing Methodology Manual)[67], is presented in this thesis. This system has been named *Consensus* and includes a wide variety of open-source testing security tools, as the OSSTMM recommends. It automates the testing process for wired and wireless networks, considering not only computers but also network devices like routers, firewalls or intrusion detection systems.

The testing process is not the only phase capable of improvement. The analysis of the data gathered in a security test is another line of research where many efforts are being dedicated. This is due to the fact that the data processing to analyze the security level of a network is difficult, especially for the high volume of data to be managed. Another reason of this complexity is related to the information collected, as it usually contains data from multiple sources with imprecise, approximate and uncertain knowledge. Thus, the following contribution presented in this thesis exposes new alternatives for the processing of this data moving closer machine learning and network security fields. As a result of the automation process of a security test and its subsequent analysis, artificial intelligence has been included as a means to promote the processing and analysis of data after a successful security test. Consequently, the application of machine learning techniques will help security experts to process the large amount of information and draw conclusions from data gathered automatically. This contribution is embodied in *Analia*, the analysis module of *Consensus* that incorporates the analyzed machine learning approaches. This module also comprises validity methods to confirm the quality of machine learning results. Moreover, it includes a contribution of a new method of validation of the results adapted to this security domain ad hoc that will be used to verify the usefulness of the system's response to the security analyst.

With the contributions detailed in this thesis, we show the effectiveness of the machine learning approach by building a framework for our model that detects vulnerabilities and analyzes the results of a security audit. These processes are automated to improve the work of security auditors and alleviate the manual tasks carried out when doing a security test over a network.

[1]European Comission CORDIS FP7 - http://cordis.europa.eu/fp7/home_en.html

1.4 Main goals

The research lines followed in this thesis have been defined mainly by the objectives of the research projects developed on the activity of research, led by the guidelines of the GRSI group in the telematics domain.

Network security is a critical aspect of any organization, company or university. It is increasingly necessary to have mechanisms in data networks to guarantee a minimum level of risk and ensure a certain level of security, with the purpose of protecting not only the devices connected to the network and its communications but also the information stored. To achieve a proper level of security it is essential to know the status of all components of a network, detect potential threats and vulnerabilities, analyze the results and correct any problems discovered in the devices. This is why the first objective of this research is the study and analysis of the techniques and methodologies used to detect vulnerabilities and perform security assessments in a network.

This thorough analysis leads to the definition of the second objective of this thesis: the proposal, design and implementation of *Consensus*, a new solution to detect vulnerabilities in a network. This new system carries out security tests over data networks on an automated way to detect security vulnerabilities and potential holes. The system stores the results of tests in order in a centralized database. This line of research is linked with the project *Consensus - Vulnerability Detection System* explained in Section 1.2.

The volume of information obtained while performing a security test is substantial, because a network is formed by a multitude of devices and many parameters and data about each of them should be gathered. In addition, it is essential to conduct security tests on a network periodically in order to maintain the level of security in a network. It is possible to detect new devices, setting changes, new vulnerabilities and other important aspects regarding network security in two consecutive assessments over the same network. Consequently, the security analyst faces a constant challenge to study the information obtained from different tests and draw appropriate conclusions. In this context, another objective of this research work is the analysis and application of machine learning techniques in the field of network security to provide mechanisms to improve information analysis obtained on the status of a network. Within this objective, the validation of data processing results from a security and telematics point of view when applying machine learning techniques has generated a new goal and, hence, new contributions have been presented in this area. The retrieval of the data processed by means of machine learning techniques has become a topic of study as well. Comprehensive conclusions should be given to security analysts so the proposed system becomes a real assistance in their daily routine. These objectives have been materialized in *Analia*, the analysis module of *Consensus* with machine learning capabilities to extract conclusions from security data. All these lines of work are related to the projects *VDS-Analia* and *MID-CBR*, explained in Section 1.2.

Accordingly, the main goals of this thesis are summarized in the following list:

- Study and analysis of the techniques and methodologies used to detect vulnerabilities and perform security assessments in a network.
- Design and implementation of the security testing contributions in a new framework called *Consensus*, a new solution to perform security assessments automatically in a network.
- Study and analysis of machine learning approaches in the field of network security to improve data analysis after a security assessment of a communications network.
- Study and analysis of validation techniques for the results of the application of machine learning approaches.

- Study and analysis of the retrieval of the data processed by means of machine learning techniques in a format understandable for the security analyst.

- Design and implementation of the contributions related to data processing with machine learning in *Analia*, an analysis module integrated in *Consensus*.

- Validation of the contributions and the framework *Consensus* and *Analia* with appropriate testing scenarios.

The integrating framework of this thesis is depicted in Figure 1.1. This graphic shows the two main blocks of contribution: the *Consensus* system and the *Analia* module. The red dotted line separates both blocks and includes the security profile associated to each block. The security tester is the profile that will benefit from *Consensus* features, whereas the security analyst is the profile that will take advantage of *Analia* facilities. These two profiles are related to the phases of a methodological security assessment [97].

On the one hand, *Consensus* will provide the resources to verify system security settings and identify system vulnerabilities of wireless and wired networks. *Consensus* will allow to view the system from the external attacker perspective by testing the network from the outside with the *Internet Testing Sensor*. It will also test the system from a malicious insider perspective from the inside with the use of the *Intranet Testing Sensor*, as it can be seen in Figure 1.1. The special characteristics of the DeMilitarized Zone (DMZ), where the public servers are located, will be solved with the use of the *DMZ Testing Sensor*. All these sensors and their integration in *Consensus* will be described exhaustively in the next chapters.

On the other hand, *Analia* is the analysis module of *Consensus* with Artificial Intelligence abilities to manage data stored in *Consensus* after a security assessment. So *Analia* will supply the mechanisms to do a structured analysis of the data collected from a security assessment and a reporting proposal. As a consequence, this module will include the machine learning techniques that will provide processed results to security analysts.

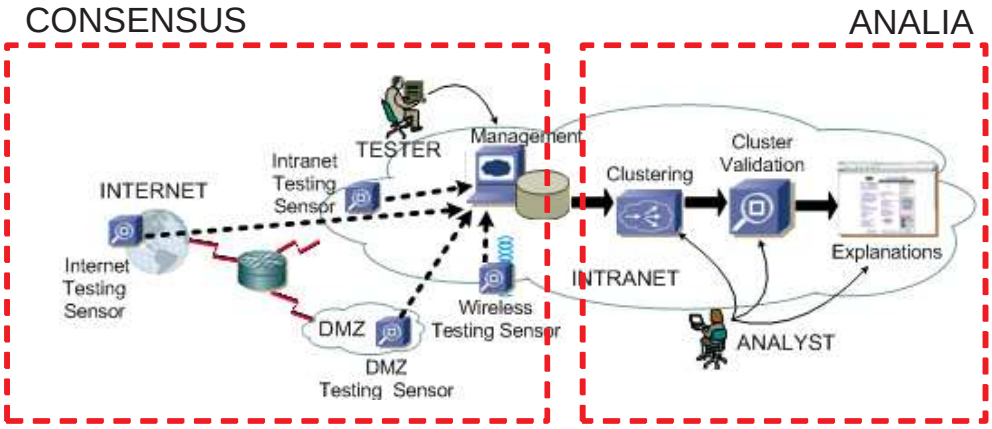


Figure 1.1: Framework of this thesis with *Consensus* and *Analia*.

The study, analysis, resolution and implementation of these objectives will be described in the following chapters. Also the various contributions for each of the research lines of this thesis will be detailed.

1.5 The thesis

This thesis summarizes all the different work and contributions developed in the data network and machine learning fields. This document is divided into three parts: (1) framework and related work; (2) contributions; (3) conclusions and future work. Chapters 1 and 2 constitute the framework and the related work part. Chapters 3, 4, 5 and 6 form part of the contributions. Finally, Chapter 7 summarizes the conclusions and future work. A brief description of the chapters included in this thesis is detailed below:

- Chapter 1 introduces the framework of this thesis and details the motivation and goals that have led us to move forward in this research area.

- Chapter 2 provides background on relevant issues related to the research lines of this thesis. A broad overview of the field of network vulnerability assessments is given and the most representative security policies, standards and methodologies are detailed. Different approaches that have been taken in other systems for automating security assessments are surveyed as well. This chapter also overviews the artificial intelligence techniques most suitable to be applied in the analysis of the information gathered in network vulnerability assessments. Unsupervised learning approaches are studied to select the most appropriate for the system. Validity techniques are analyzed and techniques to explain unsupervised learning results are addressed.

- Chapter 3 focuses on the first phase of a security assessment: the testing phase. The main goals of this phase are enumerated. The methodologies and procedures related to this phase are analyzed and several limitations are identified. The contributions to solve these limitations are presented. The framework *Consensus* is described and its components are depicted, with the aim of explaining the contributions related to the different improvements in more detail. This chapter summarizes the *Consensus* features and justifies that a new worthwhile framework for security experts has come to light.

- Chapter 4 details the experiments carried out with the framework *Consensus* to evaluate its performance in real data networks. The testing scenarios and experimentation results are detailed. These evaluations show how *Consensus* can be used to improve the daily work of security testers.

- Chapter 5 focuses on the second phase of a security assessment: the analysis phase. A discussion of the most suitable artificial intelligence techniques for this security domain and this analysis phase is included. The main steps of the clustering approach have been defined and contributions for each step have been designed and integrated in *Analia*, the analysis module of *Consensus* with artificial intelligence capabilities. This chapter explains all the different contributions in the process of unsupervised learning and how they have been used to improve the analysis of security testing results.

- Chapter 6 summarizes the experiments run with *Analia* to process data from *Consensus* framework. These experiments show how *Analia* can be used to extract conclusions from the information of security vulnerability assessments. Experimentation results are described and corroborate how *Analia* can be used to improve the daily work of security analysts.

- Chapter 7 presents the conclusions of the research and summarizes the contributions of this thesis. This chapter also shows the lines for future research.

1.6 Summary

This thesis has been developed into the research group GRSI of La Salle (URL) and has followed the guidelines of the PhD program *Tecnologies de la Informació i les Comunicacions i la seva gestió* in the same university URL. This thesis could not have succeeded without the support of the GRSI group, La Salle and the URL university.

The global context of this thesis is the telematics field. In this broad scope, the security of data networks has become the core area and, more specifically, the security assessment of data networks. In this particular area, the important benefits of machine learning, and unsupervised learning in particular, have been considered and the integration of both worlds has resulted in the final contribution presented here.

The final objective of this thesis is to define, design and implement a framework for the improvement of security assessments in data networks. This framework has automated the testing phase of a security assessment and has included machine learning techniques for the analysis of the security assessment results. The result of this framework is a new platform called *Consensus*, a distributed security testing system that establishes a guideline to deploy a complete network security test, promotes open source tools, provides a flexible implementation and adapts to user needs. This system contains *Analia*, the module where the artificial intelligent techniques to analyze testing results have been included. The framework composed of *Consensus* and *Analia* has been evaluated with a set of experiments to corroborate the success and usefulness of the whole system. These contributions have been endorsed with several publications in the fields of telematics and machine learning. All this work will be described in the next chapters at great length.

Chapter 2

Related work

The significant growth of networks and Internet has lead to increased security risks. Security vulnerabilities have become a main concern as they are constantly discovered and exploited. This chapter surveys related work about vulnerability assessment and the application of artificial intelligence techniques in this environment. The different topics covered by the thesis are detailed and the state of the art of these topics is expounded.

2.1 Introduction

Communication data networks have become a key element in the infrastructure of every corporation. Data networks are used not only to communicate devices but also to transport sensitive information, critical business applications and even to store mission-critical data. This crucial dependence on networks has generated a demand for mechanisms that assure a certain level of security.

More connectivity leads to more security risks and hence more potential vulnerabilities that could be deliberately exploited. The term vulnerability can be defined as a condition of a missing or ineffectively administered safeguard or control that allows a threat to occur with a greater impact or frequency, or both [97]. Vulnerability can also be defined as a bug or misconfiguration or special sets of circumstances that could result in an exploitation of that vulnerability, directly by an attacker or indirectly through automated attacks [113]. Related to this term, vulnerability assessment is the process of identifying and quantifying vulnerabilities in a system. Regarding network security assessment, this system can be a computer, the communications infrastructure or a whole data network. Besides, a vulnerability assessment must be an important part of any security audit.

A security audit is a systematic, measurable technical assessment of how the organization's security policy is employed at a specific site. In fact, security audits provide a measurable way to examine how secure a site really is. Information is gathered through personal interviews, vulnerability scans, examination of operating system settings and network baselines. A security audit follows a three-phase methodology. Phase one serves as a planning phase and defines the scope of the effort, the network architecture, and the existing security solutions. Phase two encompasses a number of different testing tools and activities used to verify system security settings and identify system vulnerabilities, in order to view the system from two separate perspectives, that of the external attacker, and that of a malicious insider. Finally, phase three defines a structured analysis of the collected data and a reporting approach [97]. However, due to the large magnitude of corporation networks and the different aspects to take into account, time and cost can limit the depth of a security audit. These limitations justify the automation of the maximum number of processes

involved in this task. This thesis is focused on analyzing and automating several processes related to phase two and phase three to help security auditors; phase one depends on every corporation and must be performed by auditors. Regarding phase two, automated vulnerability assessment tools can perform the scanning task on a large number of devices in a fraction of the time taken by a manual approach [83]. Nevertheless, it is also necessary to automate the processes related to the analysis of test results (phase three), as a thorough network security test generates extensive data quantities that need to be handled.

This chapter describes the related work covered by the thesis. It is divided into two main blocks: a survey of network security assessments and a study of the analysis phase of a security assessment. The former considers the definition of a security vulnerability and reviews the security policies, standards and methodologies most widely used. It describes types of security assessments. Next, it provides a study of the testing tools that have become widespread and, hence, more suitable to be included in the contribution framework. Regarding the second category, an analysis of the most suitable machine learning techniques for this scenario has been done. Different unsupervised learning approaches have been studied. Several alternatives to explain clustering results have been detailed. Finally, a study of the techniques to validate clustering results closes this related work.

2.2 Network vulnerability assessments

2.2.1 Security vulnerabilities

Security vulnerabilities have become a main concern as they are constantly discovered and exploited in computer systems and network devices. Several tools exist to analyze them from the point of view of vulnerability definition, modeling and characterization. A service offered by SANS[1] is called @RISK: The Consensus Security Alert. This system reports the new security vulnerabilities discovered during the past week and the actions that other organizations are doing to protect themselves. The NVD[2] is a security vulnerability database that integrates publicly available U.S. Government vulnerability resources. The NVD is also synchronized with the CVE[3] vulnerability list. CVE provides common names for publicly known information security vulnerabilities and exposures. Also OVAL[4] promotes open and publicly available security content. OVAL defines a language using XML for representing system configuration information and transferring results to other OVAL-compatible products. It uses CVE list as the basis for most of the OVAL definitions. On the other hand, CVE list does not give a measurement that reflects the criticism of vulnerabilities. There exists a public initiative for scoring and quantifying the impact of software vulnerabilities. This score is named Common Vulnerability Scoring System (CVSS). It represents the actual risk a given vulnerability possesses, helping analysts prioritize remediation efforts [19].

Different research projects use CVE database as a source to identify vulnerabilities. M2D2 is a model for IDS alert correlation that models vulnerabilities from CVE [90]. ALBA is an agent-aided intrusion detection tool that incorporates CVE dictionary into its ontology to represent vulnerability information [81]. CVE database has been included in systems that support development of secure software [50, 68]. Other systems use CVE information to predict the vulnerability discovery process for a program [71]. The Open Web Application Security Project (OWASP[5]) presented VulnXML, an open standard format for web application security vulnerabilities only. Unfortunately there is not much research on automated vulnerability management [112].

[1] The SANS Institute. http://www.sans.org
[2] National Vulnerability Database (NVD). http://nvd.nist.org
[3] Common Vulnerabilities and Exposures (CVE). http://http://www.cve.mitre.org
[4] Open Vulnerability and Assessment Language (OVAL). http://oval.mitre.org
[5] OWASP Project. http://www.owasp.org

2.2.2 Security policies, standards procedures and methodologies

Every device connected to a network is exposed to many different threats. While information system and network security professionals struggle to move forward with constantly evolving threats, hackers use technology that conventional security tools and services cannot cope with. Nevertheless, it is important to consider not only deliberated actions but also unintentional human activities, problems with systems or external problems that could derive in negative results like the exposure, modification or even the deletion of information or also the disruption of services.

There exists a serious problem of lack of knowledge about the solutions to establish in order to improve security in a corporation. This fact leads to an insufficient security level in many corporate networks. For example, a study performed by ASIMELEC in 2003 applied to 52 Spanish companies showed up that the implantation of security standards like ISO17799-1 was only reflected in the 53% of the corporations [44]. This study also maintains that the main cause of these poor results is due to the lack of perception of this environment and its consequences by the companies. Moreover, most companies consider that a high percentage of the security recommendations are not applicable in their context, although the main reason of the absence of applicability is due to their inexperience [44]. The international register of certificates ISMS (Information Security Management System) shows only 35 Spanish companies certified with ISO/IEC 27001:2005 in May, 2009. On the other hand, UK has 400 certified companies, Germany has 119 and the United States has 91 companies with this certification [66].
The most important security standards are the following:

- **BS7799**: standard published by the *British Standard Institute* (BSI). The standard BS 7799 (BS 7799-1) was published in 1995. It is a code of practice for information security management and it is the base to obtain a BSI accreditation that assures that the information stored in a corporate network is secure and the security of that corporation is reliable. A second part BS 7799-2 is added in 1998. Part 1 of the standard was proposed as an ISO standard and published with minor amendments as ISO/IEC 17799 on 2000. BS 7799-2:2002 is officially launched in 2002. It is re-published in June 2005 as ISO/IEC 27001:2005, as a result of the regular ISO standards update cycle.

  - **ISO/IEC 17799**: this standard establishes guidelines and general principles for initiating, implementing, maintaining and improving information security in an organization. It contains best practices of control objectives and controls of different areas like security policy, asset management, human resources security, physical and environmental security, communications and operations management, access control, information security incident management and so on. Most of the existent standards are based on this ISO and incorporate part of its recommendations.
  - **ISO-27001:2005**: this standard establishes the bases of an ISMS. It applies, as a framework, the ten operational areas defined in ISO/IEC 17799:2005. It covers all types of organizations to ensure different aspects like compliance with laws and regulations, definition of new information security management processes, and also identification and clarification of existing information security management processes. This standard follows the four-step model PDCA (*Plan-Do-Check-Act*) in order to establish, operate, monitor and constantly improve the effectiveness of a documented ISMS.

- **UNE-ISO/IEC 17799** and **UNE 71502**: UNE is the Spanish authority in charge of regulating network and information security. UNE-ISO/IEC 17799 is a best practice code for the security information management and it is based on the international standard ISO/IEC

17799. It divides the security specification in ten blocks distributed from perimetral security to the classification and control of assets. The standard UNE 71502:2004, based on BS 7799-2:2002 is approved in 2004. AENOR carries out a certification according to the Spanish standard UNE 71502:2004. This certification establishes the requirements to implement, document and evaluate an ISMS, within the risks identified by organizations according to UNE-ISO/IEC 17799:2002 standard, which develops a code of good practices for the information security management.

- **ISO/IEC TR 13335**: this standard provides guidelines for the management of information technology (IT) security. It defines general concepts about security and communication models. It also provides techniques for the management of IT security, designed to assist the implementation of IT security. Different concepts and models for information and communications technology security management are included too.

- **ISO/IEC 15408**: the main goal of this standard is to provide evaluation criteria for IT security in order to compare different models. Functional and assurance requirements can be evaluated when applying this standard.

Different methodologies for security management and system development have arisen in the last years. These methodologies are mainly focused on design, implementation and monitoring aspects. Complex systems and networks need adequate methods to formalize and validate their management and development. The main security methodologies are the following:

- **OCTAVE** (*Operationally Critical Threats, Assets and Vulnerability Evaluation*): it is a risk-based strategic assessment and planning technique for security [94]. This methodology not only focuses on technology but also on organizational risk and strategic, practice-related issues, balancing operation risk and security practices. It defines a set of self-directed activities for organizations to identify and manage their information security risks. It is an open methodology that contains three phases. Phase 1 determines what is important to the organization and what is currently being done to protect those assets. Security requirements and possible threats to each critical asset are also identified. Phase two evaluates the information infrastructure considering the network, architecture, operating system and applications. Finally, Phase three develops a protection strategy and mitigation plans to address the risks to the critical assets.

- **MEHARI** (*Methode Harmonisée d'Analyse des Risques*): it is a method for risk analysis and risk management developed by CLUSIF(*Club de la Securite des Systemes d'Information Francais*). It summarizes MARION and MELISA methods. It is compatible with other security standards like ISO/IEC 17799 and ISO 13335. The software *Risicare* is the main tool used to implement this methodology.

- **ISM3** (*Information Security Management Maturity Model*): ISM3 is a a framework for ISMS. ISM3 looks at defining levels of security that are appropriate to the business mission and render a high return on investment. It can be applied to assure business objectives are specifically tailored to security design, implementation, operations, management, procurement, and assurance processes.

- **Ebios** (*Expresión des Besoins et d'Identification des Objectifs de Sécurité*): it is used to assess and treat risks relating to information systems security. It can be used to communicate this information within the organization and to partners- Thus, it assists in the risk management process. It has been designed by DCSSI (Central Information Systems Security

Division - *Direction Centrale de la Sécurité des Systèmes d'information*), department of the French government.

- **OSSTMM** (*Open Source Security Testing Methodology Manual*): it is an open source methodology proposed by ISECOM [67]. It is a peer-reviewed methodology for performing security tests and metrics. It is divided into five sections which test information and data controls, personnel security awareness levels, fraud and social engineering control levels, computer and telecommunication networks including wireless devices, physical security access controls and security processes. This is the first open source methodology in this environment. It is considered not a corporative but an operational methodology: it does not explain how security tests should be performed. It its more focused on which items need to be tested, what to do before, during, and after a security test, and how to measure the results.

2.2.3 Security assessments

Network security is a main concern in any corporation. Constantly evolving threats force security professionals to do permanent and exhaustive controls to protect networks and detect weaknesses. Anomaly detection techniques have been widely applied to detect intrusions and vulnerabilities since the publication of the seminal report by Anderson in 1980 [5]. A correct design of a security system needs the understanding of the threats and attacks and how these may manifest themselves in audit data [5]. Consequently, testing is a basic tool to assure that enterprise security requirements are met. The security level can be also evaluated by analyzing test results and discovering how attackers can penetrate into systems and networks [34, 113].

Security assessments pursue three main goals. The first goal is to discover design and implementation flaws, as well as to identify the operations that may violate the security policies implemented in an organization. Secondly, it has to ensure that security policy reflects accurately the organization's needs. Finally, it has to evaluate the consistence between the systems documentation and how they are implemented. In fact, a complete test should include the system, communication, physical, personal, operational and also the administrative security [113].

Different security assessments can be performed. The difference between these security assessments is focused on the goal to analyze, cost and time spent in order to obtain results (see Figure 2.1, [67]). The most common security tests are the following [67]:

- **Vulnerability scanning**: it consists in discovering known security holes, flaws, softwares or techniques that take profit of any vulnerability in remote systems. Most of vulnerability scanners rely on a local database that contains all the information required to check a system for security holes in services and ports, anomalies in packet construction, and potential paths to exploitable programs or scripts. Then the scanner tries to exploit each vulnerability that is discovered. When vulnerabilities are discovered, countermeasures should be applied in the system so as to solve the problem.

- **Security scanning**: this test not only performs a vulnerability scanning but also includes manual false positive verification and network weakness identification.

- **Penetration testing**: the commonly accepted definition of penetration testing is the illegitimate acquisition of legitimate authority [49, 107, 99]. The goal of a penetration test is to gain privileged access to the network or any device. Therefore, it is based on the ability to command network facilities to operate in a different way rather than what their owners

expected them to do, to gain the full or at least substantial control of a host in a way normally reserved for trusted operations staff, or to acquire the management interface of an application [49].

- **Risk Assessment**: it is the process to determine the level of risk in a particular course of action. Risk is assumed as the possibility that some factor could damage any element of an organization. The result of this test is a report that shows assets, vulnerabilities, likelihood of damage, estimates of the costs of recovery, summaries of possible defensive measures and their costs and estimated probable savings from better protection. It is necessary to take into consideration the presence of the network, business justifications and also industry specific justifications [110].

- **Security Audit**: it usually refers to a hands-on, privileged security inspection of all the systems within a network, taking into account the analysis of operating systems and applications. This test is much more exhaustive than the previous ones.

- **Ethical Hacking**: it describes the process of attacking a network on behalf of its owners, seeking vulnerabilities that a malicious hacker could exploit. Ethical hackers report problems instead of taking advantage of them. These hackers work with clients in order to help them secure their systems. Thus, this test is not intrusive as its goal is to detect failures but not to damage any network or system [107].

- **Posture Assessment or Security Testing**: it is the process to determine that an information system is capable to protect its data and also it is capable to maintain its original functionalities [113].

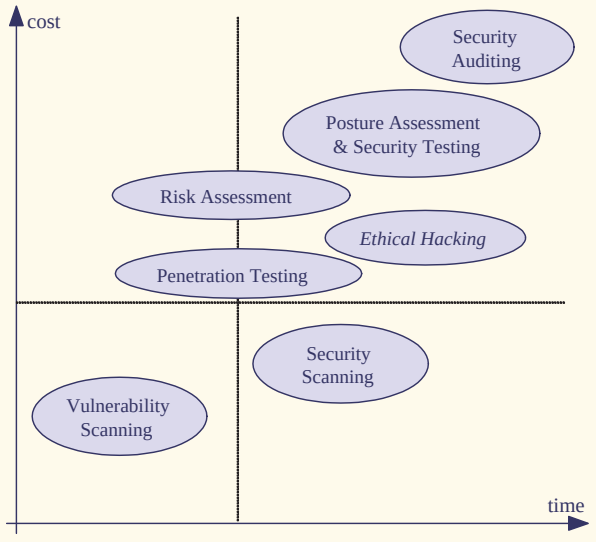


Figure 2.1: Types of system and network security assessments as based on time and cost.

2.2.4 Network security testing tools

Different vulnerability assessment tools exist in the market. Some of them are free scanners, like Nessus[6], X-Scan[7] and SARA[8]. Other commercial tools are GFI LANguard[9], Retina[10], ISS Internet Scanner[11] and SAINT[12]. The latest version of Nessus is now closed source, but it is still free without the latest plugins. The most popular and free web vulnerability scanners are Nikto[13], Paros proxy[14] and Webscarab[15], whereas WebInspect[16], Watchfire AppScan[17] and N-Stealth[18] are commercial web vulnerability scanners. Not only is cost an important matter in commercial solutions, but also proprietary methodologies that hide the internal testing process and vulnerability assessment. These types of tools do not usually implement an open source methodology for network testing. On the other hand, free or open source tools are useful when performing a test due to their inexpensive cost, flexibility and easiness of customization. However it is necessary to use different simple tools in order to perform a thorough security test, as every tool gives response to a part of the test. Thus, distinct tools are needed for system service identification, vulnerability assessment, application-specific assessment, access control testing or other phases of a network security test.

The first goal of a network vulnerability assessment is to test everything possible [97]. Different improvement proposals to achieve this objective have been found. VASE (Vulnerability Assessment Support Engine) improves safety in vulnerability testing process but only integrates Nessus as vulnerability assessment tool [54]. VAS is another system that is using Nessus as the only scanning engine for vulnerability assessments [117]. Lee et al.[75] present a system that automates port scanning by using only Nmap[19]. SAT is a tool to test and detect weakness and security holes in a computer system, but it does not detect vulnerabilities on other network devices [89]. Ferret is another host vulnerability checking tool that does not consider network vulnerabilities [104]. Another proposal is MulVAL [95], a system that uses OVAL as scanning tool. However, an OVAL scanner runs on each machine so the tool must be installed on every host that needs to be analyzed. Hence, hosts must be modified before the scan and vulnerability results could be influenced by this alteration. If the network to be tested has many devices, the previous work needed to perform the test increases dramatically. Moreover, most of the reported systems only take into account servers or computer systems and they do not consider other network devices like firewalls or routers. Furthermore wireless networks need special tests as their properties differ from wired networks. This thesis will provide a solution to automate the testing phase using complementary tools in a structured way and following an open source methodology to perform vulnerability assessments.

Protecting a network is about more than just securing servers and computer systems. The network should be also protected against attacks that target firewalls and routers. Routers provide services that are essential to the correct operation of the networks. Compromise of a router can lead to various security problems on the network served by that router, or even other networks which that router communicates with. Different auditing procedures for a network secured with a

[6]Nessus. http://www.nessus.org
[7]X-Scan. http://www.xfocus.net/projects/X-Scan
[8]SARA. http://www-arc.com/sara
[9]LANGuard. http://www.gfi.com/lannetscan
[10]Retina. http://www.eeye.com
[11]IS. http://www.iss.net
[12]SAINT. http://www.saintcorporation.com/products/vulnerability_scan/saint
[13]Nikto. http://www.cirt.net/code/nikto.shtml
[14]Paros proxy. http://www.parosproxy.org
[15]OWASP WebScarab Project. http://www.owasp.org
[16]SPI Dynamics' WebInspect. http://www.spidynamics.com
[17]Watchfire AppScan. http://www.watchfire.com/products/appscan
[18]N-Stealth. http://www.nstalker.com
[19]Nmap. http://www.insecure.org

firewall are in [79, 12]. Nessus and Nmap are the tools respectively proposed to audit manually the firewall. A case study for security audit in [78] analyzes manually the network infrastructure and its possible vulnerabilities. PBit is a tool developed to test packet filter rules only for Linux firewalls [39]. A functional approach is presented in [70], where firewall vulnerabilities are classified based on the firewall functional units to run the corresponding penetration tests for different firewall products. Another firewall testing technique based on selecting the packets instead of random testing is shown in [41]. All these solutions are focused only on testing firewalls and routers; they are not integrated in a system capable of testing all devices within a network.

The increasing popularity of wireless networks has opened organizations up to new security threats. Wireless LANs (WLANs) present unique security challenges because they are vulnerable to specialized and particular attacks. Some attacks may exploit technology weaknesses, whereas other may be focused on configuration weaknesses. Compromising only one node or introducing a malicious node may affect the viability of the entire network [34]. Typical tools for monitoring wired networks examine only network or higher abstraction layers based on the assumption that lower layers are protected by the physical security of wires. However, this assumption cannot be extrapolated to wireless networks because of the broadcast nature of such networks [77]. Currently, there exist wireless vulnerability scanners, like Aircrack[20], Kismet[21], Stumbler[22], Wellenreiter[23] or Airsnort[24]. However, different tools are needed to obtain all the information required for a reliable test. Therefore, there exists a need of a vulnerability detection system for WLANs that automates the processes. As wireless networks are usually part of a corporate network, this automated system should be integrated in a system capable of detecting vulnerabilities in the entire network. A global solution for this problem will be presented in the following sections.

Another line of work regarding vulnerability analysis is related to model network attacks, due to the fact that attacks may exploit a known or unknown vulnerability. Model checking techniques [102], attack-graph generation based on model-checking [106], graph-based search algorithms techniques [98, 3], reasoning rules using Datalog clauses [95], a framework for automatically extracting attack manifestations from log data [74], stochastic formal grammar [73] and finite-state machine modeling [22] are some approaches. However this line is out of scope of this thesis, more focused on the analysis of vulnerability reports after a network test has been performed.

2.3 Analysis of network vulnerability assessments

The last phase of a vulnerability assessment is the analysis of the assessment results to obtain a concise report that summarizes the conclusions obtained from the audit.

Regarding result analysis, systems like VASE and VAS only show Nessus results [54, 117]. MulVAL models the interaction of software bugs with system and network configurations using reasoning rules that characterize general attack methodologies. MulVAL results show violation and attack traces [95]. All these tools show vulnerability assessment results, but the analysis and process of this data is manually performed by security analysts. However, considerable data quantities are compiled after performing a network security test, and therefore a manual classification becomes an arduous work [34, 42, 81]. This is why the analysis phase needs to be improved in order to help security analysts to deal with test results. The improvements presented in this thesis introduce AI techniques in the analysis phase to handle stored data. This is why the next covered work is related to AI and networking.

[20] Aircrack. http://www.michiganwireless.org/tools/aircrack
[21] http://www.kismetwireless.net
[22] http://www.stumbler.net
[23] http://www.wellenreiter.net
[24] www.airsnort.shmoo.com

2.3.1 Artificial Intelligence and Machine Learning

The main goal of Machine Learning is to build mechanisms through which intelligent systems improve their performance over time. This paradigm refers to systems capable of the autonomous acquisition and integration of knowledge. This capacity to learn from experience, analytical observation, and other means offers increased efficiency and effectiveness.

Depending on the implemented strategy, Machine Learning techniques can be classified in supervised learning and unsupervised learning. The former estimates unknown dependencies from known input-output samples. Supervised learning classes are predefined so an expert is required to pre-classify the samples. On the other hand, unsupervised learning classes are initially unknown and need to be extracted from input data. Unsupervised learning is useful when the system does not provide data with labeled classes or any other information beyond the raw data.

2.3.2 Clustering

The analysis of information provided by network security tests requires improved methods to identify behavior patterns, recognize malicious data or unauthorized changes in a network [36]. Unsupervised learning can be applied in this security test environment, where the domain and the different classes have not been defined and no previous knowledge of network behavior and data results are required. In this sense, unsupervised learning can be used for clustering groups of tested devices with similar vulnerabilities and find features inherent to the problem.

2.3.2.1 Clustering process

The main goal of clustering is to group elements with similar attribute values into the same class. Elements from the same cluster should have a high similarity, whereas elements from different clusters should have a low similarity. In the clustering task there is no goal attribute, as classes are initially unknown and need to be discovered from the data. This means that no information related to common features in a class can be used, as classes have not been previously defined. Clustering activity usually involves the following steps:

1. Pattern representation.
2. Definition of a pattern proximity measure appropriate to the data domain.
3. Clustering.
4. Data abstraction, if needed.
5. Cluster validation, if needed.

Pattern representation refers to the number of classes, the number of available patterns, and the number, type, and scale of the features available to the clustering algorithm. A previous feature selection should be done, where the most effective subset of the original features should be selected. Secondly, pattern proximity is usually measured by a distance function defined on pairs of patterns. A broad variety of distance measures can be used. A simple distance measure like Euclidean distance can often be used to reflect dissimilarity between two patterns, whereas other similarity measures can be used to characterize the conceptual similarity between patterns. There also exists a high variety of techniques for cluster formation. For this reason, the most adequate techniques for the data domain should be also selected. Data abstraction is the process of extracting a simple and compact representation of a data set, usually in terms of cluster prototypes or representative patterns such as the centroid or the director vector. Finally, cluster validation is the assessment of a clustering procedure's output to determine whether the output is meaningful.

2.3.2.2 Clustering classification

There exist a large number of clustering methods. The choice of a clustering algorithm depends on the type of available data and on the particular purpose [57]. In general, clustering methods can be classified in the following approaches.

There are two large families of clustering techniques: *partition methods* and *hierarchic methods*. The former cluster training data into K clusters, where $K<M$ and M is the number of objects in the data set. The latter work by grouping data objects into a tree of clusters.

Clustering methods can be also classified considering the strategy used to build the clusters. *Center-based methods* minimize an objective function, usually based on distances between centroids, which indicates how good a solution is. One of the most representative partition and center-based methods is the K-means algorithm [60]. Its main drawback is the configuration of the K parameter. In this sense, the X-means [96] algorithm automatically calculates the best number of clusters for each execution. *Search-based* methods also minimize an objective function, but tend to search for the global optimal point, instead of a local global point. MOCK (Multiobjective Clustering with Automatic k-Determination) [58] is an example of a search-based clustering approach. It optimizes a partitioning with respect to multiple, complementary clustering objectives and uses a multiobjective evolutionary algorithm to perform the optimization. *Density-based methods* discover clusters with an arbitrary shape. This typically regards clusters as dense regions of objects in the data space that are separated by regions of low density, representing noise. The most popular algorithms in this category are the following: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [43], OPTICS (Ordering Points to Identify Clustering Structure) [6] and DENCLUE (DENsity-based CLUstEring) [62]. On the other hand, *Grid-based method* divides the space into a finite number of cells that form a grid structure in which all clustering operations are performed. As an advantage, this method has a constant processing time, independently of the number of data objects. In this group point up algorithms such as CLIQUE (Clustering High-Dimensional Space) [2], STING (STatistical INformation Grid) [114], and also WaveCluster [105], an algorithm that clusters using the wavelet transformation. *Model-based methods* use mathematical and probabilistic models; an example is the Autoclass algorithm [21].

Concerning artificial neural networks (NN) [13], Kohonen Map or Self-Organizing Map (SOM) [72] is a widely used unsupervised learning model. Adaptive Resonance Theory (ART) is another NN technique. There are many variations of ART available. The simplest ART network classifies an input vector into a category depending on the stored pattern it most closely resembles [16].

Different clustering techniques have been applied in the network security domain. K-means has been used to find natural grouping of similar alarm records [15, 69] and also to detect network intrusions [76, 120, 42]. Clustering has been used for network traffic classification as well[80]. SOM has already been applied as clustering method to study computer attacks [36], to detect network intrusions [37] and viruses [118], to analyze TCP traffic patterns [119], and also to detect anomalous traffic [101]. Evolutionary multiobjective clustering has been successfully applied to important real-world problems such as intrusion detection [4], formation of cluster-based sensing networks in wireless sensor networks [116] and creation of security profiles [55].

Soft Computing techniques can improve the discovery of implicit and previously unknown knowledge using a better feature extraction in a high dimensional space to cluster data [47]. An example of a Soft Computing technique is SOM [72]. However, a drawback of data clustering methods is their lack of interpretation of clustering results, as they do not explain why some elements are in the same cluster and other elements have been grouped separately. Hence a further interpretation is needed to facilitate an accurate explanation of the results.

A brief description of the clustering techniques that may enclose the best options to process data from security assessments is provided in the following sections.

2.3.2.3 *K*-means

K-means [60] is a clustering partitioning method that classifies data without having previous knowledge about this data. It requires specification of the number of partitions or clusters to obtain (K), the data set to classify and the number of iterations. It starts with a random initial partition and keeps reassigning the patterns to clusters based on the similarity between the pattern and the cluster centers until a convergence criterion is met. The cluster center is called centroid, which is the mean or weighted average of the points in the cluster.

K-means attractiveness lies in its simplicity. Firstly, the centroid for each K cluster must be initialized to random values. However, a major problem with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function value if the initial partition is not properly chosen. Several proposals have been reported in the literature to improve the process of the centroid initialization. A very simple solution may consist on executing the algorithm several times and then, choose the best solution.

Once the centroids are initialized, K-means partitions the data points into K subsets, in which all points in a given subset belong to some centroid. This means that, for each data point, the algorithm finds the centroid which is closest to the data point and associates it with this centroid. When all the points are assigned to their closest centroids, K-means recalculates centroid locations by taking, for each centroid, the mean of the points in the cluster. This iterative process finishes when the centroids locations stay fixed during an iteration or a maximum number of iterations has been reached. At the end, a partition where all data points are distributed in the different K clusters is obtained. The Algorithm 2.1 resumes the main steps of the K-means algorithm.

Algorithm 2.1: *K-means* algorithm

```
Function cluster_creation(C: Training set) is
    Randomly initialize the k centroids of the k clusters
    while changes in cluster membership or iterations<threshold do
        forall instance do
            Calculate dissimilarity between the instance and each centroid
            Assign each instance to its closest cluster centroid
            Compute the new cluster centers as the centroids of the clusters
        iterations++
    return clusters
```

For each data point, the closest centroid is selected by a similarity measure (see Equation 2.1).

$$E = \sum_{l=1}^{k} \sum_{i=1}^{n} y_{i,l} \cdot d(X_i, Q_l) \tag{2.1}$$

Where:

- 'k': number of clusters
- 'n': number of data points
- '$y_{i,l}$': function that returns whether the data point i belongs to cluster l
- 'd': dissimilarity function

The most widely used dissimilarity function is the Minkowski metric with p=2, that is the Euclidean distance. Other functions can be also implemented. The dissimilarity function is shown in Equation 2.2.

$$d_{rs} = \left\{ \sum_{j=1}^{n} |x_{rj} - y_{sj}|^p \right\}^{1/p} \quad \longrightarrow \quad d_{euclidean} = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{2.2}$$

The K-means algorithm is popular because it is easy to implement and it is often used as an exploratory data analysis tool. On the other hand, K-means suffers a major shortcoming: the number of clusters K has to be supplied by the user. In the testing security domain this is a disadvantage, as it is a completely unsupervised domain and the number of partitions to obtain from data is previously unknown. This is a lack of flexibility, but it can be solved with other K-means based algorithms that improve its weakest points like, for example, the X-means algorithm.

2.3.2.4 *X*-means

X-means [96] is an improvement of the K-means algorithm that efficiently searches the space of cluster locations and the number of clusters. It reveals the number of classes in the underlying distribution. That is much faster than using repeatedly K-means for different values of K. X-means also searches for spherical distributions and models clusters by using centroids, like K-means. It requires specification of a range of values in which the true K reasonably lies, the data set to classify and the number of iterations.

Firstly, X-means starts executing a K-means algorithm with K equal to the lower bound of the given range. It continues dividing the existing centroids where they are needed until the upper bound of K is reached. In each iteration, the algorithm splits some centroids in two and, for each new centroid, it executes a local K-means only for the data points that were associated with the original centroid. It calculates the best partition by using the BIC (Bayesian Information Criterion) criterion too. Finally, the centroid set that achieves the best score is output, as well as the value for K in the given range which has scored best. Algorithm 2.2 resumes the main steps of the X-means algorithm.

Algorithm 2.2: *X-means* algorithm

```
Function cluster_creation(C: Training set) is
    Execute K-means with k= lowerbound
    while k < K upperbound do
        Split centroids in two and execute local K-means
        k++
    return best scoring model
```

X-means preserves the advantages of K-means regarding simplicity and local-minimum convergence properties. Moreover, X-means solves some of K-means shortcomings. X-means only requires a range of values for K value, instead of the exact number of clusters. X-means also scales better computationally due to the fact that its K-means executions are performed locally. This local decision-making causes that some regions of the space tend to become active but using less data points, while other regions, where the centroids seem to have found the correct classes, do not split. However, the initial distribution of the centroids is still a critical point and a bad choice of initial centroids may have a great impact on both performance and distribution.

2.3.2.5 SOM

Kohonen maps or Self-Organizing Maps (SOM) [72] are one of the major unsupervised learning paradigms in the family of artificial neural networks [13]. SOM is classified as a Soft Computing technique that allows the management of uncertain, approximate, partial truth and complex knowledge.

The most important features of SOM are the following:

- It preserves the original topology.
- It works well even though the original space has a high number of dimensions.
- It incorporates the selection feature approach.
- Although one class has few examples they are not lost.
- It provides an easy way to show data.
- It is organized in an autonomous way to be better adjusted to uncertain and complex data.

On the other hand, the main drawback is that SOM is influenced by the order of the training samples. In addition, this approach requires several training decisions which are not trivial to solve [47].

Contrary to other neural networks approaches [13], the SOM's architecture is only composed of two layers. The input layer represents the new problem using as many neurons as input data features. Conversely, the output layer describes the new low-dimensional space using as many neurons as the maximum number of expected clusters. Although the output layer can represent any dimensionality lower than the original N-dimensional space, it usually represents a grid of two dimensions such as the example of Figure 2.2. In this case, the output layer contains $M \times M$ clusters, where each one is represented by a director vector of N dimensions (v_m). A director vector can be described as the expected value for each one of the N features. Finally, each input neuron is connected to all the output neurons. Other examples of topologies are shown in Figure 2.3.

In the training process, the models, which are represented by a set of properties using a n-dimensional vector, are iteratively fitted in order to create clusters with different properties. This process is achieved by means of updating the models using the training samples. For each training sample, a model is selected using some similarity measure. Then, the selected model vector and the neighbors models are updated to better fit to this example. Two indexes are used in this process: the learning factor α and the neighborhood factor ν. The former measures the influence of locating a new sample, whereas the latter measures the neighbor range influenced by the sample.

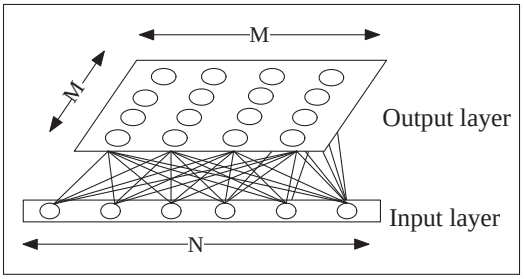


Figure 2.2: 2D-map architecture.

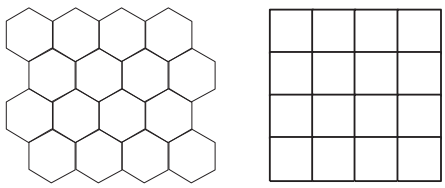


Figure 2.3: Hexagonal and rectangular topologies.

Although there are many variants of the SOM training, the procedure can be detailed in the next steps:

1. Training examples are normalized between [0..1].

2. All the components of the director vectors are randomly initialized between [0..1].

3. Given a new training example e, the most suitable cluster is computed as the minimal distance between e and each one of the director vectors. For example, the winner neuron can be the one with the lowest difference computed by the normalized Euclidean distance (see Equation 2.3).

$$similarity(e, m) = |1 - d(\overline{e}, \overline{v_m})| = \left|1 - \sqrt{\frac{\sum_{n:1..N}(e(n) - v_m(n))^2}{N}}\right| \tag{2.3}$$

4. The director vector of the winner neuron is adjusted to improve the match with new objects similar to the current one. In contrast, the director vectors of the neighbor neurons are modified to weakly represent the current example. The neighbor neurons are identified through the neighborhood factor (ν). Besides, the degree of adjustment is described by the learning factor (α). Both factors change through the training process. Initially their influence changes a lot the network while their effect is minimal at the end.

5. Steps 3 and 4 are repeated for all training examples until all the director vectors are representative enough. Usually their representation is determined by establishing a minimal error value. This value is computed as the global sum of the distance between the set of data of each cluster and its respective director vector. Nevertheless, another common criterion is to establish a maximum number of algorithm iterations.

When the training process ends, step 3 is the procedure used to map the new input example in the most suitable clusters. The training algorithm of a 2D-Kohonen map is shown in Algorithm 2.3.

Algorithm 2.3: Training algorithm of a 2D-Kohonen map.

Let T be the total number of iterations
Let E_{min} be the minimal error
Let $N_{i,j}$ be a node of a Kohonen map $K \times K$, and $\overline{N_{i,j}}$ the vector that represents its pattern
Let I_s be the instance s of the training set I represented by the vector $\overline{I_s}$
All nodes $\overline{N_{i,j}}$ are normalized between [0..1]
The learning factor $\alpha(0)$ and neighborhood factor $\nu(0)$ are initialized to 1 and K respectively
t=0
$error$=E_{min}+1
while $((t < T)\&(E_{min} < error))$ **do**
    $error$=0
    forall I_s *of the training set* I **do**
        Let N_{best} be the node best adapted to I_s following Equation 2.4
        All neighbor nodes of N_{best} get adjusted by using Equation 2.5
        $error$=$error$+ $\|\overline{I_s} - \overline{N_{best}}\|$
    error=error/$K \times K$
    Neighborhood and learning factors are recalculated (see equations 2.6 and 2.7), if $t < T_1$
    t++

$$\forall i,j,: 1 \leq i,j \leq K : \|\overline{I_s} - \overline{N_{best}}\| \leq \|\overline{I_s} - \overline{N_{i,j}}\| \tag{2.4}$$

$$\overline{N_{i,j}}(t+1) = \overline{N_{i,j}}(t) + \alpha(t) \cdot (\overline{I_s} - \overline{N_{i,j}}(t)) \tag{2.5}$$

$$\nu(t+1) = \nu(0) + (\nu(F) - \nu(0)) \cdot \frac{t}{T_1} \tag{2.6}$$

$$\alpha(t+1) = \alpha(0) + (\alpha(F) - \alpha(0)) \cdot \frac{t}{T_1} \tag{2.7}$$

$$\tag{2.8}$$

where:

Let T_1 be the total number of iterations of phase 1.

Let $\alpha(t)$ and $\nu(t)$ be the learning and neighborhood factors in the iteration t.

Let $\alpha(0)$ and $\nu(0)$ be the learning and neighborhood factors in the phase 1.

Let $\alpha(F)$ and $\nu(F)$ be the final learning and neighborhood factors.

The Kohonen maps are identified with the connectionist approach in unsupervised learning. Nevertheless, their philosophy is very similar to K-means or X-means. If we think about the centroids like neurons, K-means would resemble to a machine learning algorithm for a neural network. Consequently, the application of SOM for clustering purposes can get the benefits of K-means and X-means algorithms, while maintaining its own benefits due to its soft computing capabilities.

To conclude, SOM is a smart technique to identify hidden and complex relationships between elements and also to identify the most relevant features thanks to its knowledge discovery and its soft computing capabilities. However, an important disadvantage of SOM is that this approach is influenced by the order of the training samples. Moreover, it requires several training decisions which are not trivial to solve. For example, the training must be stopped at the right time. If training continues for too long, it may result in overlearning, that means that the neural network extracts too much information from the individual cases forgetting the relevant information of the general case. So the correct number of training iterations will have to be configured in the system to avoid overlearning.

2.3.2.6 Autoclass

Autoclass is an unsupervised model-based learning algorithm that uses mathematical and probabilistic models [21]. More specifically, it is an unsupervised Bayesian classification system that seeks a maximum posterior probability classification. In the Autoclass approach, class membership is expressed probabilistically; every item is considered to have a certain probability to belong to each of the possible classes. As a result, Autoclass differs from partition algorithms, like K-means or X-means, in the sense that the former does not calculate distances between samples, but uses probabilistic theory. It can automatically find the set of classes that is maximally probable with respect to input patterns in the training set.

Automatic Autoclass only requires specification of the maximum number of iterations and the data set to classify, due to the fact that the algorithm automatically calculates the optimal number of clusters for the partition obtained. As an alternative, if the number of clusters to obtain is previously known, it can be specified as an input parameter and the partition will contain as many clusters as desired.

The output of Autoclass is the most probable classification given the data and prior expectations. It does not give a set of case classes, with a class for each unique case. So, instead of assigning instances to certain classes, it uses a weighted assignment, weighting on the probability of a class membership.

The main steps of Autoclass are the following:

1. It assigns the ownership probability to a class of elements.
2. For a certain data set, it calculates:
   - The most probable values of the parameters given a certain probability distribution.
   - The most probable probability distribution: number of classes and alternative models, independently of the parameters.

Autoclass basic model is based on the classical mixture distribution, that is a two part model. The first part gives the interclass mixture probability that an instance X_i is a member of a class C_j, independently of any other knowledge about the instance. Each class C_j is then modeled by a class probability distribution function, giving the probability of observing the instance attribute values conditional on the assumption that the instance X_i belongs to the class C_j.

An advantage of the Autoclass approach is that it provides an automatic choice of the number of clusters. Another difference with the other studied techniques is that Autoclass does not assign each input pattern to a single cluster, but it returns a cluster membership probability for each of the obtained clusters. On the other hand, Autoclass provides models for nominal and several types of numerical data, but not for ordinal data. Another restriction of this approach is that data is limited for which instances can be represented as ordered vectors of attribute values.

2.3.2.7 Evolutionary multiobjective clustering

MOCK is an unsupervised learning technique employed to evolve data clusterings that optimize two complementary objectives: the overall deviation and the connectivity [59]. The former is related to cluster compactness and the latter is based on connectedness of clusters. It uses the PESA-II algorithm. PESA-II evolves a set of solutions, where each one defines a possible clustering configuration. In what follows, the knowledge representation, the process organization to obtain the Pareto set of solutions, and the method to recover the best solution among the ones in the Pareto set are briefly explained.

Representation. The system evolves a population of individuals of size N, where each individual represents a cluster structure for the problem. The individual is represented in a vector of n integers: $x_1, x_2, \ldots, x_n$. Then, x_i indicates that instance i is connected to instance x_i, that is, that they belong to the same cluster.

Process organization. The result of the algorithm is a Pareto set of solutions, that is, a set of individuals for which it does not exist any other individual in the population that dominates them[25]. The population is evolved as follows. We first initialize the population with an individual that represents the minimum spanning tree (MST) constructed from the undirected, fully connected labeled graph that represents the Euclidean distance between each pair of examples. In addition, we also create $N - 1$ individuals that progressively remove the links with highest distance from the original MST. Then, the population iteratively goes through a process of selection, crossover, and mutation as described in [59].

Selection of the best solution. After evolving a set of non-dominated solutions, we use the following criterion to recover the best solution. The system returns the best solutions that optimize the couple connectivity/deviation.

[25] In multiobjective algorithms, a solution x dominates another solution y if all the objectives of x are better than the corresponding objectives of y.

2.3.3 Explanation of clustering results

The idea of explaining results comes from initial expert systems where explanations consisted on the chain of rules used to reach the final result (for instance, MYCIN [100]). This kind of explanations is specific for rule-based systems and then it is necessary to define some other kind of representations depending on the representation used by the system. Thus, Case-Based Reasoning systems [1] commonly use as explanation the set of cases that are the most similar to a given problem. Explanation-based learning methods [35] give as explanation of a result the generalization of the reasoning followed to prove that an example belongs to a class. Conceptual clustering assembles entities into a single cluster not because of their pairwise similarity, but because together they represent a concept from a predefined set of concepts [86]. Consequently, this approach not only clusters entities, but also provides concept descriptions of the obtained clusters [86]. However, conceptual entities must be defined a priori [108] in conceptual clustering. This requirement is not always possible in several domains, like the network security domain, where analysts do not have a complete knowledge about the problem domain and can only deal with data instances themselves. A different method should be applied in this environment to give an explanation of clustering results to security analysts.

Different viewpoint of explanations can be found in the literature. On the one hand, several authors have oriented their research to analyze how explanations have to change either the usage of the system [14, 17], or the user's expertise [85]. On the other hand, authors have focused their research on interacting with the user to refine the proposed solution, proposal commonly followed by recommender systems [52, 84]. In particular, authors analyze which aspects are relevant to explain the result. Concerning CBR systems, the common explanation of showing the user the set of more similar cases seems not be the best one when cases have a complicated structure [38, 85]. For that reason, an explanation should make explicit the contribution of each attribute to the classification problem [93], and even both similarities and differences are useful to explain the result [85]. In this context, an explanation scheme based on similarities for classification problems in CBR was proposed [9] using the Anti-unification technique [8].

How the explanation scheme proposed in [9] could be adapted to explain the results of clustering methods? The main difference among methods applied to classification problems and clustering methods is that in latter methods the solution classes are not known. This means that no information related to common features in a class can be used, as classes have not been previously defined. As a result, the contribution described in next sections presents an explanation scheme that shows the features that all the elements of a cluster have in common to justify their membership to the same cluster.

2.3.4 Validation of clustering results

Clustering is the process of dividing a set of given patterns into groups, or clusters, so that all patterns in the same group are similar to each other, while patterns from different groups are dissimilar. The correctness of clustering algorithm results is verified using appropriate criteria and techniques. Since clustering algorithms define clusters that are not known *a priori*, irrespective of the clustering methods, the final partition of data requires some kind of evaluation [56].

The clustering system can be assessed by an expert, or by a particular automated procedure. The former option is time consuming and requires human participation, thus limiting the automation of the system. The latter option is based on applying cluster validation functions that quantitatively evaluate clustering results. This option is named cluster validity or cluster validation. If cluster results can be automatically validated, then the efforts of security experts become reduced and the whole process can be automated.

Two main parameters are defined to evaluate clustering results [11]. These parameters are the following:

- Compactness: the elements of a cluster should have a high degree of similarity, that is to say, should be as close to each other as possible.
- Separation: the clusters themselves should be widely spaced. This distance between two different clusters can be measured between the closest members of the clusters, between the most distant members or between the centers of the clusters.

Cluster validity methods can be classified in three main groups [111]:

- External criteria: this approach validates whether the points of the data set are randomly structured or not, based on statistical tests.
- Internal criteria: this approach evaluates clustering results using only quantities and features inherent to the dataset. The dataset vectors and proximity matrix are used to assess the results.
- Relative criteria: this approach assesses clustering results by comparing them with other clustering results, coming from the same clustering algorithm but configured with different parameters. It chooses the best clustering scheme of a set of defined schemes according to a pre-specified criterion.

Several validity indices proposed in literature for the relative criteria approach have been analyzed to be included in *Analia* [10, 53, 56]. These indices are the following: *C-index* [63], *Davies-Boudin* (DB) [33], *Dunn* [40] and *Silhouette* [103]. A description of these indices is detailed in next sections.

2.3.4.1 C-index

The validity index *C-index* [63] only measures the compactness of each cluster, but it does not consider the separation between clusters. This index is defined by Equation 2.9 for a single cluster.

$$C(c) = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{2.9}$$

Where:

- 'S': sum of distances over all pairs of patterns from the same cluster
- 'S_{min}': sum of the smallest distances if all pairs of patterns are considered
- 'S_{max}': sum of the largest distances out of all pairs

Hence a small value of C indicates a good clustering.

2.3.4.2 Davies-Boudin index

Davies-Boudin (DB) index [33] attempts to identify compact and well separated clusters. This index is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The scatter within the cluster Q_i is computed as Equation 2.10.

$$S_n(Q_i) = \frac{1}{\|c_i\|} \sum_{x \subset C_i} d(x, v_i) \tag{2.10}$$

Where:

- 'x': element in cluster i
- '$||c_i||$': number of instances of cluster i
- 'v_i': center of cluster i
- '$d(x, v_i)$' distance between the element x and the mean of cluster i

The separation between two clusters is calculated by the function S(Q_i, Q_j), which measures the distance between clusters centers. The global formula of DB index is shown in Equation 2.11.

$$DB(c) = \frac{1}{n}\sum_{i=1}^{n} max_{i\neq j}\left\{\frac{S_n(Q_i) + S_n(Q_j)}{S(Q_i, Q_j)}\right\} \tag{2.11}$$

Where:

- 'n': number of clusters
- '$S_n(Q_i)$': average distance of all elements from the cluster i to their cluster center
- '$S(Q_i, Q_j)$': distance between the center of cluster i and the center of cluster j

By the definition of Equation 2.11, it can be seen that DB index is the average similarity between each cluster and its most similar one. It is desirable for the clusters to have the minimum possible similarity to each other; hence the ratio is small if the clusters are compact and far from each other. Consequently, clusterings that minimize DB are preferred.

2.3.4.3 Dunn index

Dunn index also aims to identify sets of clusters that are compact and well separated [40]. The Dunn's validation index is shown in Equation 2.12.

$$D(c) = min_{1\leq i\leq n}\left\{min_{1\leq j\leq n, j\neq i}\left\{\frac{d(c_i, c_j)}{max\left\{diam(c_k)_{i\leq k\leq n}\right\}}\right\}\right\} \tag{2.12}$$

Where:

- 'n': number of clusters
- 'c_i': cluster i
- '$d(c_i, c_j)$': distance between clusters c_i and c_j
- 'diam(c_k)' intracluster distance or diameter of cluster k

If the dataset contains compact and well-separated clusters, the distance between the clusters is expected to be large and the diameter of the clusters is expected to be small. Thus, large values of Dunn's validity index correspond to good cluster partitions. However, the computation of this index requires a considerable amount of time and it is sensitive to noise in the data set [56].

2.3.4.4 Silhouette index

Silhouette index is a confidence indicator on the membership of a sample in its cluster [103]. The formula to calculate the *Silhouette width* for each sample is shown in Equation 2.13.

$$S(i) = \frac{b(i) - a(i)}{max\left\{a(i), b(i)\right\}} \tag{2.13}$$

Where:

- '$a(i)$': average distance between the i^{th} sample and all of the samples included in its cluster
- '$b(i)$': minimum average distance between the i^{th} sample and all of the samples clustered in other cluster X_k

It is followed from Equation 2.13 that $-1 \leq S(i) \leq 1$. If *S(i)* is close to 1, it means that, that sample has been assigned to an appropriate cluster. If *S(i)* is about zero, it suggests that, that sample could also be assigned to another closest cluster as well, and the sample lies equally far away from both clusters. If *S(i)* is close to -1, it means that, that sample has been misclassified and is merely somewhere in between the clusters.

S(i) is assessing only the classification validity of a sample i. Then, it is possible to calculate a *Silhouette* index for a given cluster j, which characterizes the heterogeneity and isolation properties of such a cluster. This index is S_j and its formula is shown in Equation 2.14.

$$S_j = \frac{1}{m} \sum_{i=1}^{m} S(i) \tag{2.14}$$

Where:

- 'm': number of samples in cluster j
- 'S(i)': *Silhouette* index for the sample i

Finally, a global *Silhouette* index (GS) for all the partition can be calculated. The overall average *Silhouette* width for the entire partition is simply the average of the *S(i)* for all objects in the whole dataset. This global *Silhouette* index is shown in Equation 2.15. The largest value of GS indicates the best clustering solution.

$$GS = \frac{1}{c} \sum_{j=1}^{c} S_j \tag{2.15}$$

Where:

- 'c': number of clusters in the dataset
- 'S_j' *Silhouette* index for the cluster j

Part II

Contributions

Chapter 3

Consensus: Improving testing phase in a vulnerability assessment

In today's digital world, enterprises find it difficult to protect the confidential information, their networks and devices while maintaining a public Internet presence. Administrators of networks are expected to be far more knowledgeable than what was expected of them previously, as different technologies, network devices, applications and operating systems need to be managed as a whole. This rise in complexity makes it difficult for administrators to stay on top of security threats. Thus security experts can provide an unbiased and accurate analysis of the security infrastructure of an organization. On the other hand, even security experts need mechanisms to ease their tasks, due to the fact that security audits consume time and resources. Vulnerability assessments play an important role in any security audit. Improvements can be performed in testing and analysis phases of a vulnerability assessment. This chapter details the contributions of this thesis to the testing phase of a vulnerability assessment, where important data is gathered from a network and its devices.

3.1 Introduction

The complexity of computing systems and networks, the rapid increase in viruses and attacks, the unmonitored mobile users and telecommuters, and the dependence of corporations on Internet are just some of the reasons why networks are easier to break into than ever before. Besides, the tools used by hackers are becoming simpler and more accessible each day. The need for vulnerability assessment is not just to confirm the security of a network and its devices; it also stems for the concern that a network might not be adequately protected from the exponential number of threats. The design of a security monitoring surveillance system needs the understanding of threats and attacks that can be performed against systems and how these threats may manifest themselves in audit data [5]. Moreover, a network security analysis must coordinate different sources of information to support effective security models [34]. These sources of information should be distributed through all the network in order to monitor and manage the maximum number of communications.

A correct protection of a network needs to perform periodically security tests so as to control devices and services, and also identify possible vulnerabilities. Security assessments should not be performed just once. Testing should be recurring throughout the year and security experts should not rely on just one testing firm. In this sense, network security experts are looking for solutions that provide thorough, flexible, quantifiable, repeatable and consistent network security tests.

Although there exists information about testing networks and about vulnerability or penetration testing, there is no standard that regularizes the involved procedures. The success of a test comes when having a solid methodical plan, not by running several tools at random. Thus

a plan that details exactly what, when and how to do is necessary in any vulnerability test. The discussion of security policies, standards and methodologies (see Section 3.2.1) will highlight the OSSTMM [67] as the appropriate methodology for security auditing. This manual outlines a global comprehension of the many security issues to be checked to perform a valid security test.

A testing methodology like OSSTMM helps to verify network security reliably, but it can consume a great amount of time due to the exhaustive tests and the security knowledge needed to test, gather and analyze results. Time and budget constraints can have a negative influence on the depth and extent of a security assessment. The more time and resources needed to plan, learn about the testing tools, install them, configure them with the right parameters, wait for the results, process multiple results files from the different sources, identify the meaningful data and obtain valuable security conclusions, the less profitable that security assessment becomes. Security experts should focus their efforts on the critical issues, being constantly in touch with new threats and exposures, and they should not spend the most part of their resources on routine and mechanical tasks. This is why the automation of these tasks becomes an imperative to alleviate the daily work and, consequently, enrich security assessments results, due to the fact that less time, resources, budget and specific knowledge about every single tool will be needed.

The main goal of this thesis is to improve the processes related to a security assessment, so this automation becomes a main topic of interest to solve. Moreover, the solution presented in this thesis to this main issue is a new framework called *Consensus*, a new approach for automating a security assessment following the OSSTMM methodology. The *Consensus* system will provide solutions not only for the testing phase but also for the analysis phase of a security test. Although the various contributions for each phase will be detailed in the next sections, they are all integrated in this new system *Consensus*, providing a unique platform as a whole to ease security management.

This chapter proposes different improvements to automate the processes associated to a vulnerability assessment based on the OSSTMM. The contributions will be focused on the testing phase, with the intention of giving solutions to the different demands that security testers may pose. These contributions will be built in *Consensus*, the framework of this thesis. The design and implementation of *Consensus* will be detailed in this chapter. This research work has been partially supported by the Spanish grant FIT 360000-2004-81 as part of the project *'Consensus - Sistema de Detección de Vulnerabilidades'*.

3.2 Consensus analysis and considerations

The design of a new framework for security testing implies a previous analysis of the existent security policies, standards and methodologies related to security. This analysis must conclude with the best methodology to be followed by this new framework. When a conclusion has been reached, a further analysis of the parts of the selected methodology that can be automated will be performed. Also the best methods to automate each phase will be considered. The decisions of this discussion section will influence on the design and implementation of this new security platform, *Consensus*.

3.2.1 Discussion of security policies, standards and methodologies

The most representative security standards and methodologies have been enumerated in Chapter 2. Their main characteristics are summarized in tables 3.1 and 3.2. Table 3.1 summarizes the main characteristics of the most important security standards. For each security standard, it includes the year of publication, the country of origin, the basis of the standard, the organization that provides the standard, its main orientation, whether it is certifiable and open-source or not, whether it complies with other standards and, also, if there exists any support tool. As another point of view,

Table 3.2 summarizes the same features explained before, but only for security methodologies, not standards. This classification will be used to discuss the best options to include in a security system like *Consensus*.

Standards	**BS7799-2:2002**	**ISO/IEC 17799**	**ISO 13335**	**ISO 15408**	**UNE 71502**	**UNE-ISO/IEC 17799**	**ISO 27001**
Year	2002	2000	1996	1998	2004	2002	2005
Country	U.K.	U.K.	—	—	Spain	Spain	—
Precedents	BS7799-2	BS7799-1	—	—	BS7799:2-2002	ISO/IEC 17799	BS 7799-2:2002
Type of standard	Standard. Process oriented. Organizational	ISO standard. Best-practices oriented. Organizational Operational	ISO standard	ISO standard	UNE standard. Process oriented. Organizational	UNE standard. Best practices oriented. Organizational	ISO standard. Process oriented. Organizational
Certifiable	Yes	No	Yes	No	Yes	No	Yes
Open source	No	No	No	No	No	No	No
Compliance with other standards	—	ISO9000, ISO13335, ISO15408	ISO 7498	—	ISO9000, ISO14000	ISO9000, ISO14000	ISO9001, ISO14001
Tools	—	Risicare	—	—	—	—	—

Table 3.1: Comparison of security standards.

Methodologies	**OCTAVE**	**MEHARI**	**ISM3**	**EBIOS**	**OSSTMM**
Year	2001	1996	2004	2002	2001
Country	USA	France	Spain	France	USA
Precedents	—	MARION MELISA	—	—	—
Type of methodology	Risk management oriented. Organizational	Best-practices oriented. Organizational	Process oriented. Organizational	Risk management oriented. Organizational	Operational
Certifiable	Yes	No	No	No	Yes
Open source	Yes	Yes	Yes	Yes	Yes
Compliance with other standards	—	ISO 17799 ISO13335	ISO 17799, BS7799, COBIT, ITIL, CMMI	NF X 50-151 ITSEC, CC ISO/IEC 15408	ITIL, ISO/IEC 17799, GAO, NIST, LOPD, LSSICE
Tools	—	Risicare	—	EBIOS Software	—

Table 3.2: Comparison of security methodologies.

Tables 3.1 and 3.2 show that the standards that best satisfy the security goals of an organization are, on the one hand, ISO/IEC 17799 and ISO-27001:2005 as organizational standards and, on the other hand, OSSTMM as an operational methodology. The former, although they are not open-source, they comply with other standards and they are constantly revised since their origin and organizations can obtain certifications about their compliance. Regarding OSSTTMM, this is the only operational open-source methodology found in the literature. It also complies with many European and American standards, like ITIL, *IT Baseline Protection Manual*, ISO/IEC 17799, FISCAM, standards defined by NIST and so on. It also complies with Spanish laws, like the LOPD (*Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal*) and LSSICE (*Ley*

34/2002 de Servicios de la Sociedad de Información y Comercio Electrónico).

None of the standards shown in Table 3.1 are open source. In contrast, all the security methodologies of Table 3.2 are open source. This is why the discussion is based only on methodologies, as they do not need any authorization to be implemented. Regarding security methodologies, it has been shown that OSSTMM is the most suitable to be selected: it describes how to perform a security test as an operational methodology, it complies with many other standards and laws, it is certifiable and it is open source. In conclusion, OSSTMM has been selected to be the security methodology of the new framework *Consensus*.

3.2.2 Discussion of security assessments

One of the main reasons to perform security tests is to identify existent vulnerabilities with the aim of eliminating them. A correct security system needs to check periodically the state of the network and its devices to detect new vulnerabilities. The CERT Coordination Center catalogued only 417 vulnerabilities in 1999. This number was duplicated in 2000 (1090 vulnerabilities), was duplicated again in 2001 (2437 vulnerabilities) and doubled again in 2002 (4129 vulnerabilities). In 2006 doubled again its total number, achieving 8064 vulnerabilities. At the end of the third quarter of 2008, 6058 new vulnerabilities have been already catalogued [18].

After having analyzed the different security standards, it has been shown that there is no standard that specifies how to perform a security test. Besides, OSSTMM is becoming more accepted as an standard de facto, so that security testers do not have to reinvent the wheel when designing a methodology for security auditing [115]. Therefore *Consensus*, the security system proposed in this thesis, will be based on the OSSTMM by automating the different procedures addressed in the methodology, in order to simplify test executions. The OSSTMM manual outlines a global comprehension of the many security issues to be checked to perform a valid security test. It focuses on the technical details of exactly which items need to be tested, what to do before, during, and after a security test, and how to measure the results [67]. A security test may be considered an OSSTMM test if it is quantifiable, consistent, repeatable, valid beyond the time frame, thorough and compliant to individual and local laws and the right to privacy. This fact allows an objective comparison between two different security tests.

The OSSTMM addresses different areas of a security assessment. They are related to information security, social engineering, Internet technology security, communications security not only wired but also wireless, and also physical security. However, not all these sections can be automated. Some sections are referred to elements not directly related to networks or computers. For example, the Physical security section is in charge of reviewing the physical perimeter security measures of a corporation, like building locations, fences, gates, lights, wiring, access to closets or racks and so on. Another example that cannot be automated is the Process Security section, which demands to query personnel about information of the organization. Other elements whose automation would become very costly are, for example, the monitorization in Internet of all the information related to a corporation.

Finally, the most suitable section to become automated is the Internet Technology Security. Each module of this section has been analyzed in further detail to study the viability of its automation. Also the Wireless Security section has been selected to be automated. In conclusion, Table 3.3 shows, for each OSSTMM section, which modules are suitable to be automated. These modules will be included in *Consensus* and their automation will improve the time and resources needed to perform a security test.

Section	Module	Automation
Information Security		No
Process Security		No
Internet Technology Security	*Logistics and control*	No
	Network Surveying	Yes
	Internet Application Testing	Yes
	Routing	Yes
	Trusted Systems	Yes
	Access Control Testing	Yes
	IDS testing	Yes
	Containment Measures Testing	Yes
	Password cracking	Yes
	DoS Testing	Yes
	Security Policy Review	No
Communication Security		No
Wireless Security		Yes
Physical Security		No

Table 3.3: Sections susceptible of automation of the OSSTMM methodology.

3.2.3 Discussion of security tools

Table 3.4 presents a comparison between the tools referred in Chapter 2. This comparison considers different features regarding the application architecture, the possible platforms or operating systems, whether they include simple tools or not, the types of test to perform, and even if they are open source. It also checks whether the tools are compatible with any security standard or methodology detailed before. All these tools, except from *Consensus* which is the new contribution of this thesis, are also included in the study conducted by Insecure.org, where a ranking with the best 100 security tools has been published [65].

Security experts can choose between commercial and open-source security tools. The former provide a powerful and complete architecture to test a wide variety of security aspects, but they usually follow some proprietary methodologies, difficult to customize to an open testing methodology. Their cost also tends to be high. The latter are inexpensive but most of them test only particular security issues. In this sense, different open-source tools need to be combined so as to obtain all the required information. Independently of their cost, the tests can be performed on the same individual host where the software is installed or on remote hosts. Tests that monitor individual hosts mainly report performance statistics such as CPU utilization, page fault rates, and network statistics. Alternatively, tests to remote hosts can report port scanning, operating system and application detection, and vulnerability assessment [75]. So many different data can be obtained from the tested devices, becoming a more complex task the collection of this data. Anyway, not only gathering information about devices can become a burdensome task but also its storage and the subsequent analysis. The analysis of the information obtained from different sources will be further analyzed in Chapter 5 with the intention of providing new solutions to this problem.

In summary, none of the analyzed tools totally complies with security standards or methodologies. Security testers do not have any tool to evaluate methodically network security using open source tools and methodologies. This is why one of the main goals of this thesis is to provide a consistent vulnerability assessment system that includes different security tools and follows an open source methodology like OSSTMM. This proposed system is called *Consensus*. This system can automatically execute a consistent security vulnerability assessment from different parts of the network. This automated assessment can be applied to different technologies, not only wired but also wireless networks. These tests can be performed from inside the corporate network or

Tool	Architecture	Platform	Simple tools	Type of test	Open source	Compatibility
AATools	Single system	*Windows*	No information	Port scanner, proxy analyzer, email verifier, links analyzer, network and process monitor	No	No
Internet Scanner	Single system	*Windows*	No information	Port scanner, application-level vulnerability assessment	No	No
LANGuard	Single system	*Windows*	No information	Port scanner, CGI scanner, vulnerability assessment	No	No
Nessus	Single system	*Linux Windows*	—	Vulnerability and port assessment	Yes	No
Nmap	Single system	*Linux Windows*	—	Port scanner	Yes	No
OSSIM	Distributed. Integrated modules	*Linux*	Snort, Nessus, Ntop, Snortcenter, Acid, Riskmeter, Spade, RRD, Nmap, P0f, Arpwatch. Open-source and simple tools	Network monitor, event detection, logs analysis	Yes	No
Retina	Single system	*Windows*	No information	Network discovery, port scan, vulnerability assessment	No	No
Consensus	Distributed. Independent modules	*Linux*	Dhcping, Nmap, Xprobe2, Amap, Nessus, Nikto, Irpas, Ftester, John the Ripper... Open-source and simple tools	Network discovery, port scan, OS detection, global vulnerability assessment, web, ftp and mail vulnerability assessment, router and firewall test, IDS test and DoS test	Yes	OSSTMM

Table 3.4: Comparison of security tools.

from outside, having the same access privileges a possible hacker could manage. Next sections will describe the *Consensus* design, implementation and also the different experiments performed on various networks. *Consensus* has been also included in Table 3.4, in order to describe its main characteristics to compare it with the rest of the existent tools. According to Table 3.4, *Consensus* has a distributed architecture, composed of different modules, all based on *Linux* as operating system. It includes many simple tools so many different type of tests can be performed over a network. All the tools included are open-source and this new system follows the OSSTMM methodology. The architecture of *Consensus* and the automation of the different tests are broadly detailed in the next sections.

3.3 Consensus architecture

This section describes the design and architecture of the *Consensus* system, the new framework to provide a platform that improves the processes related to security assessments. It also details the different contributions that *Consensus* brings in the testing phase of a vulnerability assessment.

3.3.1 Global architecture of *Consensus*

Consensus is the new platform that agglutinates the different proposals of this thesis to improve the security testing mechanisms [30]. It can be defined as a vulnerability detection system with a distributed architecture, useful to simplify security test executions based on the OSSTMM. As stated before, OSSTMM is an open source methodology to verify reliably the network security, but its application can consume a great amount of human and machine hours due to the exhaustive tests and the security knowledge needed to test, collect and analyze all the gathered data. Thus automation for certain processes to help security professionals is desired. In this sense, *Consensus* provides a new framework that reduces the time and resources spent during a manual security test. *Consensus* contains only open source security tools as the OSSTMM philosophy recommends.

The main goal of *Consensus* is the automation of the maximum number of processes specified in the OSSTMM methodology to perform a security test. This automation will minimize the amount of time spent to perform a methodological security test and, simultaneously, it will provide the same amount of active risk prevention in real-time that a manual test would provide, due to the fact that the methodology followed is the same in both situations. This is why *Consensus* will have to automate not only the execution of the different security tools but also the subsequent data processing and storage. Also the analysis of these results will demand to be improved, but this fact will be further analyzed in Chapter 5.

Several issues need to be addressed when designing a new security assessment system that intends to accomplish the different requirements of security experts. This system must be modular, scalable, adaptable to different networks, manageable, compliant with some security methodology and, obviously secure. All these requirements have determined the design goals of *Consensus*, which are the following:

- The system has to perform vulnerability assessments in a network and over the Internet in real time.
- The use of open source tools is mandatory to automate the processes of a vulnerability assessment.
- The system requires a web interface to configure and control the security tests, and also to configure and monitor the different probes distributed along the network.
- Highly scalable system to allow multiple probes and network technologies.
- Processing and storage of vulnerability assessment results in a centralized database for their subsequent analysis.
- Visualization of testing results by using a web interface.
- Use of a customized operating system improving its security features.
- Use of a proprietary protocol for the communication and management between the central system and the different probes.
- The system has to accept new modules and tool updates for improved test integrations.

A thorough security test must take into account very different aspects of a network. So an automated proposal for a vulnerability assessment must be as modular as possible in order to incorporate new technologies, tools and updates whenever they come out. This requirement has lead to design *Consensus* as a system composed of 8 different modules [25, 29, 30, 31]. These modules can be classified in two main groups: an administrative group that includes the *Management*, *Database* and *Analysis* modules and, on the other hand, an operational group that comprises the *Internet*, *Intranet*, *DeMilitarized Zone* (DMZ) and *Wireless* modules, which are the probes or sensors that perform the vulnerability assessments. The administrative group can be centralized in the same device or can be distributed and they will be the control modules used by security auditors. The operational group will be used depending on the network technology used to test and the scope of the vulnerability assessment.

All the probes integrate a set of simple but reliable open-source security tools. The probes perform automated security tests following the instructions sent by the management module. This module contains a web interface used by the security tester to control all the system and test the network and its devices. Different vulnerability assessments can be performed, depending on the extent and the type of devices to test. Once a test has finished, security analysts can visualize test results with the web interface included in the analysis module. It can be seen that only seven modules have been already mentioned. The last module is the *Base System*. In fact, it is not a separated module as it provides a common and customized platform for the rest of the modules. All the other modules are built upon the *Base System* and are able to perform only the system calls provided by the customized operating system.

Figure 3.1 shows the global architecture of *Consensus* and the relationship between its modules, as explained before. All the sensors or probes communicate only with the *Management Module*, controlled by the security tester. Also security analysts work with the *Management Module* to access to testing results in the *Analysis Module*. Next sections will detail the characteristics of the different modules.

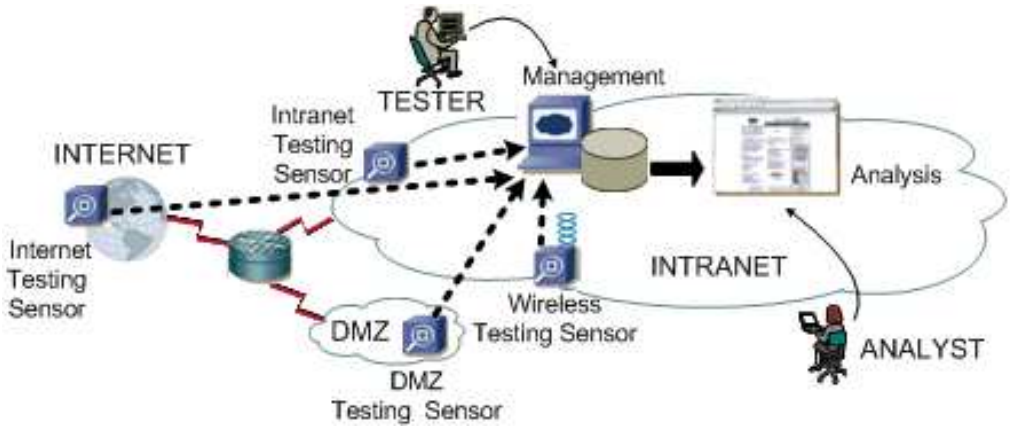


Figure 3.1: Global architecture of *Consensus*.

3.3.2 Base System Module

Consensus is a security tool that must deal with classified data available only to authorized people. This system collects data regarding security aspects of a network, so a maximum level of security is needed to assure confidentiality, integrity and authenticity, essential requirements of any security system. The *Base System Module* will provide a secure framework for the whole system, not only supplying a secure physical access but also a secure remote access.

The *Base System Module* is not shown in Figure 3.1, due to to the fact that it is not installed in a physical device different from the other modules. As its name states, it is the basis for the rest of the modules and every other module from *Consensus* will run over it. The main purpose of the *Base System Module* is to provide a common platform for configurations, security management and patching, standardization of the system environment and stability. The rest of the modules and probes use their features to perform any action, and hence every single action is controlled by the base system.

Figure 3.2 shows the block diagram of this module. The low level includes the core operating system customized with the minimum number of required plugins to have a stable, configurable and efficient system in a small size. This basic operating system must be secured in order to provide authentication and encryption for all the communications among the different modules and also to secure the access to the system. The higher level provides the communication primitives to be used by the other modules, a block for patches and updates not only for the operating system but also for the testing tools included in the probes, and another block for managing the licenses of the system in case they are activated.

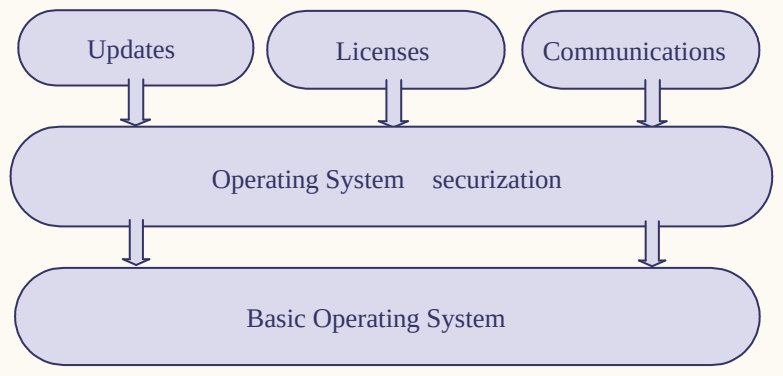


Figure 3.2: Diagram of the Base System Module.

3.3.3 Management Module

The *Management Module* is the core of *Consensus*. Its design, the defined communications protocol and its graphical interface allows the interaction between this module and the different testing probes. It also allows the global administration performed by the security expert. The purpose of the *Management Module* is to ensure standardized updates and configuration changes from a single platform. It is centrally aware of all subsystems and tests as they occur to provide a real-time overview, regardless the location of the different testing probes or sensors. All the rest of *Consensus* modules are administered from this module by using its web interface.

The *Management Module* is divided into two main blocks: the *Communications block* and the *User Interface block*. The former is in charge of the communications between the *Management Module* and the testing probes. It uses the primitives provided by the *Base System* and applies the communications protocol designed for this purpose. The latter contains a graphical user interface to control the whole system. This management module includes probe and user administration, configuration of new security tests, a remote control of the probes and a continuous display of the probes status, the testing logs and the data analysis system. These two main blocks are directly related, due to the fact that the information provided by the communications block regarding the testing and probe status must be shown to the user in real time. There is also another minor block that contains a small local database. Information related to the granted users that

can access to the system, the configured sensors to be activated when testing and, the different programmed tests and their logs are stored in this database. Figure 3.3 shows the block diagram of the *Management Module*. It includes the *Communications Block*, the *User interface block* and also the local database.

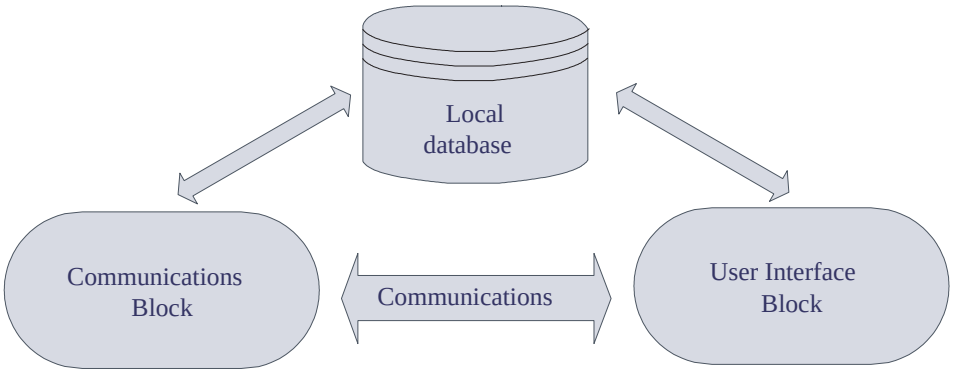


Figure 3.3: Block diagram of the *Management Module*.

The communications protocol follows a master and a slave model. The *Management Module* is the master and has unidirectional control over the rest of the modules, so the probes are the slaves that receive orders from the master. Once the relationship between modules is established, the direction of control is always from the *Management Module* to the Intranet, Internet, DMZ and Wireless probes, the Database and the Analysis modules. The communications protocol is detailed in Section 3.3.7. With this architecture, the *Management Module* maintains always the control of the whole system and is aware of the status of the rest of the modules in real time.

3.3.4 Database Module

The *Database Module* is focused on the creation, maintenance and administration of the database used to store all the information obtained after a vulnerability assessment. This information is provided by the different testing probes after performing a security test on a network. Information obtained from several testing tools is parsed and integrated in the same database, offering a unique data source to generate security reports. The *Analysis Module* will process the data stored in this database to show testing results to analysts. This module incorporates also a communications interface to interact with the *Management Module*. In fact, testing probes will not directly deal with the *Database Module*. Probes will send the testing results to the *Management Module* in an XML file and this module will be in charge of sending all this information to the *Database Module*.

The *Database Module* is composed of four blocks. The *Synchronism Block* focuses on synchronizing the database with the *Management Module*. This block notifies the *Input Block* that a new testing results file has arrived and that it must be processed. Then this *Input Block* will parser the XML file and will insert testing results in the database in a structured way, by using the primitives provided by the *Synthesis Block*. This last block includes the functions library used to insert, read, delete or modify the information stored in the database. Finally, the *Database Block* contains the entity-relationship model and the different tables with data registers. Figure 3.4 shows the block diagram of the *Database Module*, that includes the Synchronism, Input, Synthesis and Database blocks.

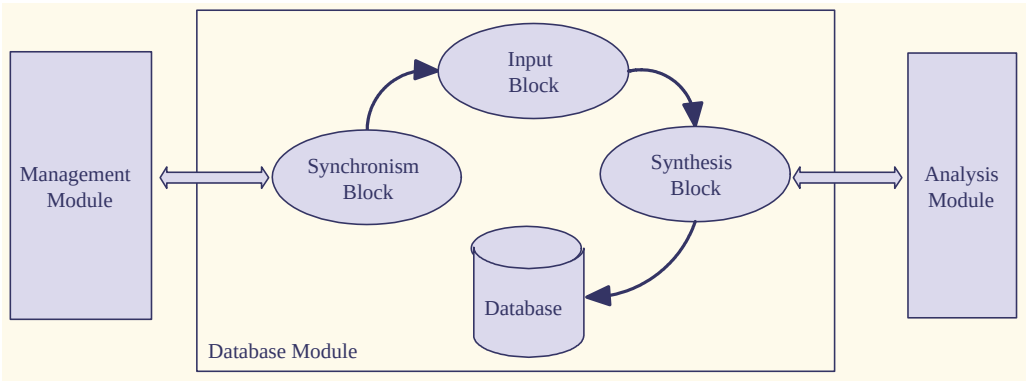


Figure 3.4: Block diagram of the *Database Module*.

3.3.5 Analysis Module

The main goal of the *Analysis Module* is to show the security test results that the different testing probes have collected. It uses the information stored in the *Database Module* to generate the security reports. It has to correlate all the data from all the sensors and provide reports for the security analysts. This module is directly related to the last phase of any security audit, which defines a structured analysis of the collected data and a reporting approach [97]. All the improvements proposed for this module in this thesis will be detailed in Chapter 5.

3.3.6 Testing Modules

The design and implementation of *Consensus* is totally scalable as it allows the configuration of independent probes or sensors, depending on the type of network or technology to test. As a contribution of this thesis, the testing modules that have been designed and incorporated in *Consensus* are the following: *Internet*, *Intranet*, *DMZ* and *Wireless* [25, 29, 31]. These four modules share similar properties, due to the fact that their structure, architecture and operational mode is the same. All of them are slaves and act in accordance with the *Management Module* rules. Their main differences consist in their location in the network, the type of tests they can perform and the open-source security tools that they include.

The block diagram of a *Testing Module* is composed of four parts: the communications block, the control block, the results block and the test block. The block diagram of a *Testing Module* and its interaction with the rest of the system is shown in Figure 3.5.

All the testing modules need a communications interface to interact with the *Management Module*, which will send them the different commands and a configuration file to inform the probes of the different testing parameters. This interface is part of the *Communications Block*. It uses the communications protocol to control the different testing modules and also to send or receive all the information related to the test (see Section 3.3.7). When the testing module has finished a test, it formats all the information and sends the results file to the *Management Module*, which will insert the data in the *Database Module*. The *Control Block* in a *Testing Module* manages all the parameters and supervises how the different tests are performed. It also controls any problem it may arise during the tests and informs the *Management Module* about the incident. The probes also incorporate a testing engine in the *Test Block*, which is the main responsible to execute the security test. This engine uses the information obtained from the *Control Block* and runs several security tools with the parameters configured in the *Management Module* interface. These tests are executed following the OSSTMM methodology. When the security test has finished, the output of the different testing tools is processed in the *Results Block* and an XML file is generated. This

file will be sent to the *Control Block* and the *Management Module* will be able to collect it to store all the results.

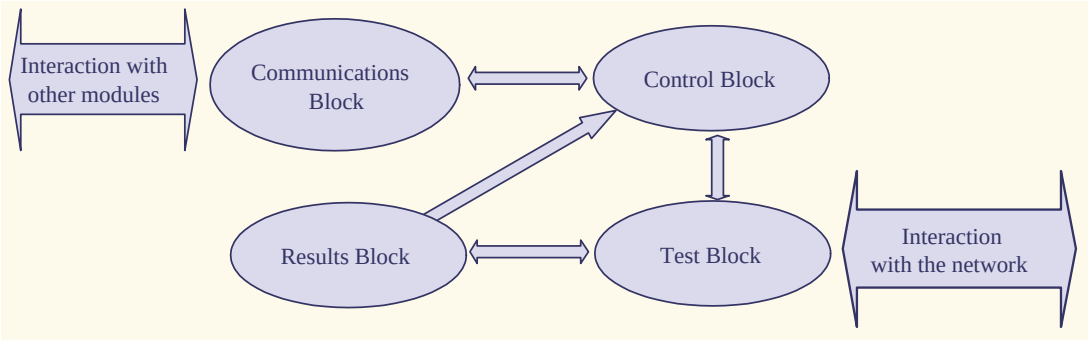


Figure 3.5: Block diagram of a testing module.

The *Internet Testing Module*, based on the OSSTMM, will provide a thorough security test of all Internet-visible systems. This test is performed from a non-privileged environment, using the same privileges a potential hacker could obtain. It will check for all currently-known vulnerabilities and also report information leaks and weaknesses in network design or stability. This module has to encrypt results during all the test process, as the information is traveling through a public and non-protected network, like Internet.

The purpose of the *DMZ Testing Module* is similar to the *Internet Module* but it is more focused on the security of the perimetral network. The DMZ contains and exposes an organization's external services to a larger, untrusted network, usually the Internet. The DMZ adds an additional layer of security to a corporate network, in the sense that an external hacker only has access to the equipment in the DMZ, rather than the whole of the network. The DMZ usually hosts the services provided to users outside of the corporate network. Therefore, the *DMZ Testing Module* will focus its efforts on checking the security of these services by using specialized testing tools.

The goal of the *Intranet Testing Module* is to provide tests to verify internal network security, information leaks and network stability. These tests are performed from a privileged environment, as the probes are located inside the internal network. So local devices not visible from Internet will be also checked. Then, these tests will be able to ensure that proper encryption and handling of private, sensitive and confidential material will be maintained too.

Finally, the purpose of the *Wireless Testing Module* is to provide automatic tests to verify wireless network security. This module will analyze the security level from the wireless network point of view and will test whether this part of the network can compromise the security of the whole corporative network. A more detailed explanation of the different testing probes is exposed in sections 3.4 and 3.5.

3.3.7 The communications protocol

The communications protocol used by the different modules to communicate with each other has been specially designed for *Consensus*. It is based on the 3-way handshake mechanism used by the TCP protocol [109]. The IETF proposal called '*Intrusion Detection Exchange Format* (IDWG)' [64] has been also analyzed. Although the main goal of IDWG could resemble *Consensus* communications protocol, the main problem of the former is that it sends a defined and fixed data structure for every message. On the other hand, Consensus needs more information to be sent on these messages, so flexibility becomes a main requirement. Also IDWG plans sending one message per every computer scanned, but this fact can be a problem in a network with many computers

because too many messages will be sent. This is why a new protocol has been designed for this purpose, including the advantages of IDWG.

The *Management Module* communicates with all the other modules by using different messages. These messages permit verifying whether the testing probes are disabled, waiting for instructions, testing or have finished a test and have results ready to be processed. A particular message has been designed to do an emergency stop of a testing probe when there is any problem in the test process.

Besides sending different messages, the different modules have to exchange several files. When the *Management Module* wants to start a certain test, it sends a file to the correspondent probe with all the parameters to set up the test. When the probe has finished the test, it has to notify the end to the *Management Module*. Then, the management entity will get the results file from the testing module. The protocol used to exchange files is SCP (Secure CoPy). It is an open source Secure Shell File Transfer Protocol, useful to provide security during the exchange process.

Figure 3.6 shows the process of message exchange between the testing, management and database modules. The communications follow a master and a slave pattern. Consequently, the *Management Module* is always the responsible for starting any dialog with the rest of the modules and waits for their response. The leadership is the basis of the *Management Module* that will poll the other modules to get information about the sensor status, the testing phase, the results file and also it will order the insertion, deletion or reading of testing results in the database.

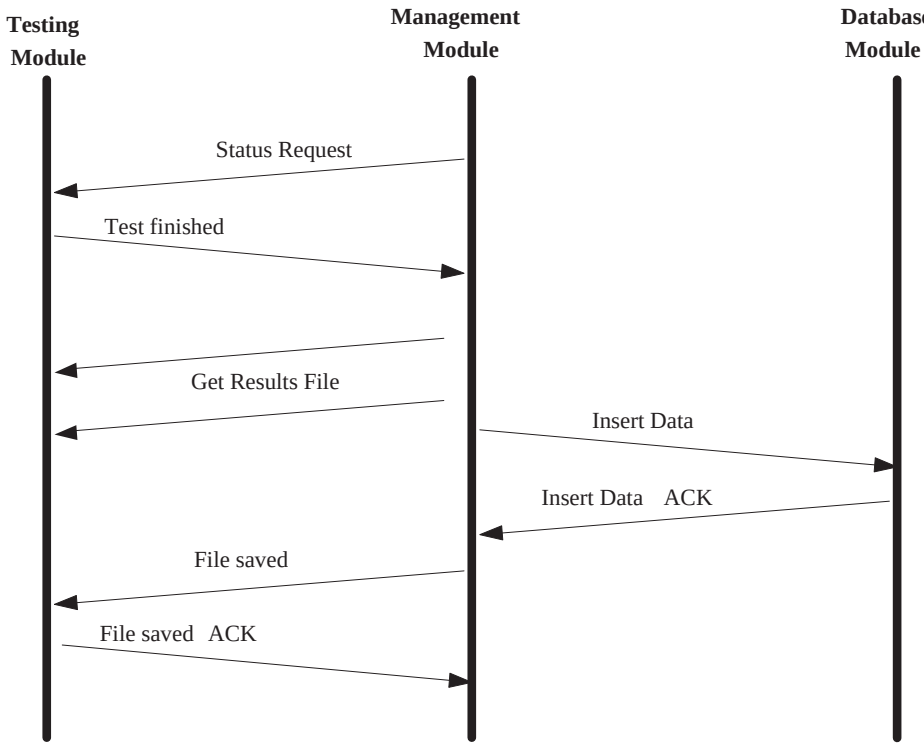


Figure 3.6: Message exchange.

3.4 Wired vulnerability assessment

This section describes the main aspects of a vulnerability assessment performed over a wired network. It also details the contributions of this thesis in this field with the aim of providing an original improvement in security assessments. These contributions are integrated in the *Consensus* system in the form of the Intranet, Internet and DMZ modules. All these modules help constructing an entire framework to automate security tests and minimize the efforts, knowledge and time spent when performing this type of tests.

3.4.1 Wired security requirements

The open nature of the Internet makes it increasingly important for corporations to pay attention to the security of their networks. As organizations move more of their business functions to the public network, they need to take precautions to ensure that attackers do not compromise their data or that their data does not end up being seen by the wrong people. Security threats may come from inside, outside or from transit points of the network. The traditional security paradigm of "assuming connections inside the perimeter are safe and connections outside the perimeter are suspect" is not nearly enough to protect a corporation's digital assets. Today's networks need security that extends from servers to all its end points, whether they are inside or outside the corporate perimeter. This is why security assessments should be performed not only from the outside to the inside, but also taking into account the inside perspective.

In Section 3.2.1, the discussion regarding testing methodologies has concluded in selecting OSSTMM as the most adequate methodology to perform security tests. Besides, the sections most suitable to be automated have been enumerated in Table 3.3. However, the functions of every section and the selected tools may vary depending on the point of view from which the test is being generated. Therefore, several solutions to cover the different possibilities where a security test may be executed from will be explained in the next section. These solutions are built in the Intranet, Internet and DMZ modules, in order to distinguish the separate approaches [29, 31]. All these modules are integrated in *Consensus*, the framework presented in this thesis. The main goal of all these modules is to help in the testing task of a security assessment. The automation of the involved processes will improve security work with a simplified administration and testing interface. Moreover, it will optimize the time needed for gathering information to discover vulnerabilities within a network and its connected devices.

The automation of security tests in wired networks will be mainly focused on the network, transport and application layers of the IP architecture, as they are the layers more susceptible to vulnerabilities. Many data can be obtained from these layers and different tools will be needed, as no unique open-source tool exists that provides all this information. It will be required to find the reachable systems to be tested, to identify the available services and applications, to scan the possible vulnerabilities of these services, to test the routers, firewalls and Intrusion Detection Systems (IDS), to test the containment measures that handle malicious programs, to validate password strength of important systems and also to deploy denial of service (DoS) tests when necessary. However, several restrictions must be considered in security testing. For example, a security assessment performed from a non-privileged environment like Internet can not include DoS tests. Packets are traveling through a public network and could become dangerous traffic for intermediate devices that could be affected or even worse, blocked by these packets. Also restrictions regarding the available bandwidth must be taken into account, due to the fact that Internet bandwidth and access links to corporate networks may considerably differ from internal bandwidth and thus, the traffic generated for a security assessment must be specially controlled. These requirements and other considerations will be further explained in the next section for the purpose of justifying the necessity of the Intranet, Internet and DMZ testing modules of *Consensus*.

3.4.2 Wired testing modules in Consensus

Consensus is a new approach for automating a vulnerability detection system applying a testing methodology like OSSTMM. The core of this automation is consolidated by the different testing modules. They supply this automation by implementing several OSSTMM phases with the use of different security tools in diverse network technologies. These differences will be reflected in the design of the Intranet, Internet and DMZ testing modules, regarding wired networks, and in the design of the Wireless module, regarding wireless networks. The former is detailed in this section, whereas the latter is described in Section 3.5.

Depending on the origin of the security assessment and the main objectives to test, different modules are going to be presented. The wired testing modules designed and implemented in *Consensus* are the following:

- **Intranet Testing Module**: Test probe that performs security assessments from a privileged environment, the internal network. This assessment may have visibility to all the devices of a corporate network and less restrictions to obtain security information about them. It focuses on the network from an internal hacker's perspective [29].

- **Internet Testing Module**: Test probe that performs security assessments from the outside to the inside. This assessment needs to cross the boundary between the Internet outside and the LAN inside the corporation, and this boundary may filter traffic or impose traffic restrictions [31].

- **DMZ Testing Module**: Test probe that performs security assessments over the public or private servers installed in the DMZ network. This network should be accessible not only from the internal network, but also from the Internet. Specific tests for this kind of servers will be included in this testing module.

All the testing modules share the same block diagram, so as to provide a similar and common design, easy to implement when new testing probes are needed. This design has been already described in Figure 3.5. Their Communications, Control and Results blocks are all the same and contain the same functions and primitives. The main difference between the wired and the wireless probes is the *Test Block*. In the wired probes, this block has been divided into ten states, that are associated with the different automated OSSTMM phases: Network surveying, System service identification, Vulnerability scanning, Application testing, Router testing, Firewall testing, IDS testing, Containment measures, Password cracking and Denial of Service. This is the most complete flow diagram but, depending on the restrictions of the testing modules, some of these phases will be skipped. The flow diagram of the *Test block* in a wired testing module is shown in Figure 3.7 and the functions of every state are the following:

- **Network surveying**: The main goal of this phase is to find the number of reachable systems to be tested. The configuration file sent by the *Management Module* specifies the scope to test, but maybe some devices are off-line. The detected on-line devices are stored in the *Host_list* file, to be used for the rest of the testing phases. Devices in the configuration file that have not been detected on-line will be discarded. In addition, this phase discovers the domain names and server names related to the tested devices.

- **System service identification**: This phase enumerates live or accessible services. It is necessary to perform port scanning, the invasive probing of system ports on the transport and network level. This port scanning identifies the open and filtered ports for every device listed in the *Host_list* file. A filtered port means that a firewall, filter or some other network

obstacle is blocking the port and preventing the system from determining whether it is open. Ports not listed as open or filtered are considered closed ports. Once open ports have been identified, it is necessary to conduct an active examination of the application listening behind the service and this information is stored in the *Applications* file. After service identification, the next step is to identify the system through the active probing for responses that can distinguish its operating system (OS) and version level. This information is stored in the *Operating system* file. The combination of these files creates a summary file to be used in the next phases.

- **Vulnerability scanning**: This phase identifies and verifies weaknesses, misconfigurations and vulnerabilities within the services and devices detected in previous phases. Vulnerability scanning employs software that seeks out security flaws based on a database of known flaws, testing systems for the occurrence of these flaws. For this purpose, a currently popular scanner like *Nessus* and different exploits have been integrated into this phase. Depending on the detected services in the previous phases, different testing plugins are loaded to perform the vulnerability scanning.

- **Application testing**: This phase tests the most important services more exhaustively, like Web, DHCP, DNS, FTP and Mail services. Specific tools are integrated in order to find security bugs that are particular of these type of services.

- **Router testing**: This phase is specifically reserved to test the boundary router or other internal routers. This phase has to find the router type and features implemented and also information on the router as a service and a system. Information about the implemented routing protocols will be gathered as well. This phase is detailed in Section 3.6.

- **Firewall testing**: This phase is specifically reserved to test the network firewall. The firewall controls the flow of traffic between the enterprise network, the DMZ, and the Internet. It operates on a security policy and uses Access Control Lists (ACLs). This phase is designed to assure that only permitted traffic will be allowed in the network, and all the rest should be denied. This phase is detailed in Section 3.6.

- **IDS Testing**: This phase is focused on the performance and sensitivity of an IDS. Evasion and insertion techniques will be used in this phase. Results will have to be compared with the IDS logs and alerts, to facilitate the verification of the performed tests.

- **Containment measures**: This phase evaluates the handling of malicious programs, exploits, viruses, trojans and other egressions by the network. This phase has focused mainly on testing the detection of these kind of programs in the e-mail server, as this is usually their main entrance in a network.

- **Password cracking**: This phase tries to discover weak passwords. Password cracking is the process of validating password strength through the use of automated password recovery tools that expose either the application of weak cryptographic algorithms, incorrect implementation of cryptographic algorithms, or weak passwords due to human factors. This phase needs a passwords file with the purpose of checking its strength. The results of this phase will include a file with the cracked and uncracked passwords.

- **Denial of Service**: This phase performs DoS attacks when they are explicitly required. DoS objectives must be planned carefully as they can affect the whole network. This is an optional phase and it returns the device response to DoS attacks.

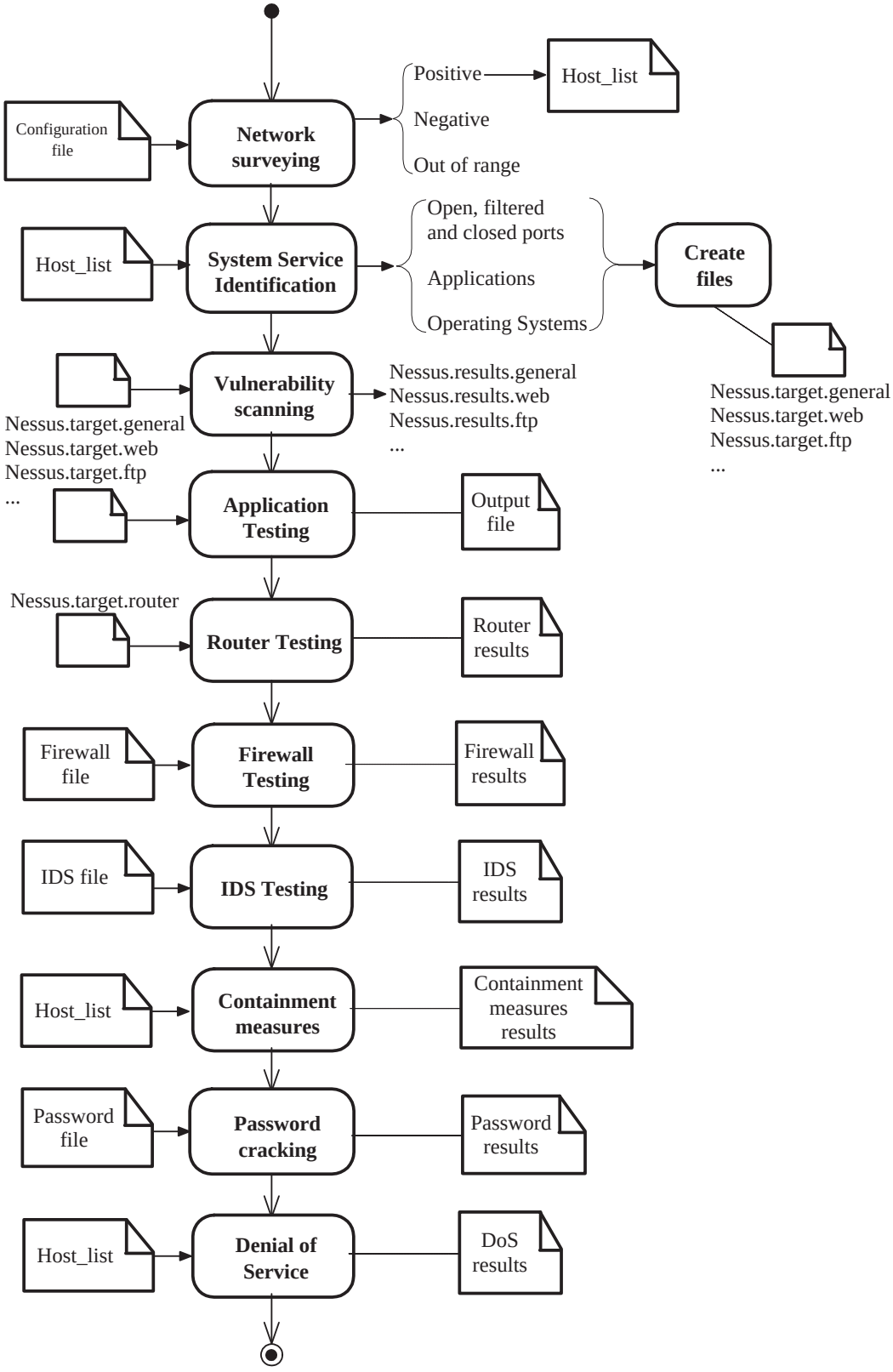


Figure 3.7: Flow diagram of the Wired Test block.

Several security testing utilities have been analyzed so as to select the most suitable procedure to automate the phases explained before. Different tools have been selected to gather all the information required. These tools must comply with several requirements:

- Linux-based tools
- Open-source tools
- Tools with command line interface
- Tools with a manageable output

Table 3.5 shows the selected tools, the data collected by each tool and the OSSTMM phase where the tools are integrated. The execution of these tools has been automated in the different probes or sensors. The data acquired may vary depending on the security tool.

Data acquired	**Tools**	**Testing phase**
Host name	*Nessus*	Network surveying
Operating system	*Nmap, Xprobe2*	System service identification
Port state	*Nmap*	System service identification
Service offered in opened ports	*Nmap, THC-Amap*	System service identification
General vulnerabilities	*Nessus*	Vulnerability research
Specific vulnerabilities	*Nessus, Nikto, md-webscan*	Application testing
Routing information	*Irpas*	Router Testing
Filtering rules	*Ftester*	Firewall testing
Answers to malicious code	*Email, Netcat*	Containment Measures Testing
IDS responses	*Nmap, Nessus, Nikto*	IDS testing
Weak passwords	*John the Ripper*	Password cracking
Response to DoS	*Unicornscan, Juno*	Denial of Service

Table 3.5: Tools for testing in *Consensus* wired probes.

When the probe has finished the test, all the partial files obtained in the different testing states of the flow diagram (see Figure 3.7) are compiled in a single results file. Once the probe signals the *Management Module* that the security assessment has ended, the data transfer and the data insertion processes follow the phases already explained in Figure 3.6. Afterwards, all the information gathered by the different automated tools is stored in the *Consensus* database to be analyzed.

All the wired testing probes share the main design features explained before. In contrast, each testing module has also its own distinguishing characteristics, that will be explained straightaway. A summary containing a comparison between the different testing modules is shown in Table 3.6. This table presents the similarities and differences between the different wired probes (Internet, Intranet and DMZ) and the wireless testing module. The Similarities column contains the features common to all the probes, whereas the Differences column details the features specific to each probe.

The *Internet Module* must comply with several restrictions. This module is installed outside the network to test and, therefore, the security assessment is performed from a non-privileged environment like Internet [31]. In this sense, the *Internet Module* has to use the existent open doors of the network and gain access to the corporate network like any other application. Thus no modifications can be performed over the existent routers or firewalls, as these modifications would alter the main purpose of the security test. Another requirement is concerned with the available bandwidth. External connections usually have less bandwidth than internal connections. So automated tools of this module will be demanded to spend less bandwidth in order not to

Probe	Differences	Similarities
Internet	BW restrictions Not physically accessed by the network administrator	Due to OSTMM, same issues to test regardless of device location Almost the same testing tools Vulnerability severity decision is not probes responsibility
Intranet	Large number of devices to test Test DHCP servers	
DMZ	Test servers performance Passive communications scanning	
Wireless	OSI levels 1 and 2 Specific parameters not present in other modules	

Table 3.6: *Consensus* probes description.

saturate the links. The last requirement is related to the kind of tests that can be performed over Internet. The system must be very cautious with the packets sent across Internet. It is a public network and the intermediate devices should not be affected by the content or purpose of the testing packets. This is why the DoS phase is prohibited in this environment and will be deactivated. Also the Router and IDS testing phases will be partially deactivated. The intermediate routers that may separate the Internet Module from the peripheral router of the network should not be tested as they are part of Internet, but not of the testing objective. Even the Password cracking phase will not be performed. This phase needs the passwords file as input, and this is a confidential file that should not travel across a public network.

The *Intranet Module* performs security assessments from a privileged environment, the internal network. Consequently, limitations regarding bandwidth will not be so restrictive as in the case of the *Internet Module*. On the other hand, the *Intranet probe* should be located in the infrastructure of the corporate network. So it is very important to minimize the impact and the amount of traffic generated when doing tests to many devices [29]. In this module, the critical factor is related to the number of devices to test. A corporate network may contain a large number of devices and many information will be gathered and stored. All the phases shown in Figure 3.7 are implemented in this module. It also includes tests to DHCP servers, that are usually located in the internal network. Communications between the Intranet and the Management modules are easier comparing with other probes, as both modules will be located in the same network or in a close distance. Anyway, communications between these modules will be sent encrypted like any other probe, with the intention of assuring secured communications.

The *DMZ Module* performs security assessments in the perimetrical network. This is a non-privileged environment already located in the corporate network. The DMZ is used to host the public services of the corporation and can be accessed not only from Internet but also from the internal network. So the DMZ probe is in charge of testing these public services and the devices that host these applications. All the phases shown in Figure 3.7, adapted to existing services, can be executed in this module. It can scan passive communications too.

3.5 Wireless vulnerability assessment

This section outlines the distinctive features of a vulnerability assessment performed over a wireless network. It also describes the design and architecture of the modules of *Consensus* related to a wireless security test. Finally, it details the different contributions that *Consensus* provides in the testing phase of a vulnerability assessment on wireless environments.

3.5.1 Wireless security requirements

Wireless networks use radio waves for communication. Although this kind of networks provide the benefits of easy mobility and installation, it makes it easy for anyone outside a building to gain access to a corporate network. Threats to wireless networks include passive monitoring, unauthorized access to applications, and even operation disruption of the network. Most wireless devices are wireless network-ready. End users or network managers often do not change the default settings or implement only standard Wired Equivalent Privacy (WEP) security, which is not an optimal solution for secure wireless networks. Thus network managers need to provide end users with freedom and mobility without offering intruders access to the network or the information sent and received on the wireless network.

A security test of a wireless network also needs to follow some methodology in order to correctly perform the test. Like in wired networks, some processes can be also automated to help security analysts in their task. Different data must be collected and there is no unique tool that can provide all the information. Thus a new environment that integrates this automation with improved features and presents results in a structured way is highly demanded. This environment will be designed and implemented in the *Wireless Module* of *Consensus*, and it will be presented as a contribution of this thesis in next sections [25].

Another requirement directly related to wireless networks is about their coverage area. When performing a security test over a IEEE 802.11 wireless network, it is necessary to be located in its coverage area in order to obtain the information that travels through the air. Then, if a corporation has different wireless networks to cover all its space, the tester will have to move around to detect the different areas and many information will be obtained. So an automation of several processes will be also very useful. In this situation, a previous analysis of the strategical points to locate the testing probes should be necessary to improve the whole process. On the other hand, other wireless technologies only have a very small coverage area or require a direct line of sight, like bluetooth or infrared, so a security test becomes more difficult to automate. This is why this section and the contributions presented are mainly focused on the IEEE 802.11 technology, the Wireless LANs (WLANs).

The IEEE 802.11 standard includes the physical layer and the data link layer, equivalent to layer 1 and 2 of the OSI architecture, respectively. This means that the main differences between LANs and WLANs are located in these two layers, and the rest of the layers share the same properties and behavior. Therefore, the *Consensus* modules used to test a IEEE 802.11 network will be the following:

- *Wireless Module*: module in charge of testing layers 1 and 2 of the OSI architecture.
- *Intranet Module*: module in charge of testing the wireless network from the layer 3 to the application layer.

The methodology selected to perform a security test on wireless networks will be the same used for testing wired networks, the OSSTMM. More specifically, tests will be focused on the IEEE 802.11 standard, due to the fact that it is the most used wireless technology of today.

The OSSTMM methodology states that the results to be obtained when performing a test over a wireless network are the following:

- Verify that the organization has an adequate security policy that addresses the use of wireless technology.
- Determine the level of physical access controls to access points and devices controlling them.
- Perform an inventory of all wireless devices in the network.
- Identify the IP range and the use of DHCP, authentication and encryption of wireless networks.
- Evaluate configuration of wireless devices and determine hardware and software faults that may facilitate attacks.
- Evaluate wireless clients.

Finally, after analyzing the OSSTMM for a wireless security test, the most susceptible OSSTMM phases to be automated in the *Wireless module* are the following: Visibility assessment, Accessibility testing, Circumvention research, Access cracking and Survivability testing. These phases and the procedures implemented to automate the testing process will be detailed in the next section.

3.5.2 Wireless testing module in Consensus

The main goal of the contribution presented in this section is to automate the processes involved in a security test over a IEEE 802.11 wireless network. This contribution is materialized in the *Wireless Module* [25], whose design and implementation is explained afterwards.

The *Wireless module* has to gather all the information related to the physical and data link layers of the wireless devices. This information must include data regarding the detected Access Points (AP) and their location, channel usage, emitting power, encryption and authentication methods, the detected SSID and so forth. This module also has to detect the network addresses of the APs, due to the fact that this information will be provided to the *Intranet Module* to keep on with the rest of the security assessment. The *Wireless Module* has been integrated with *Consensus*, in order to obtain a powerful security testing system capable of dealing with different technologies.

The *Wireless Module* has been designed following the same guidelines of the rest of the testing modules. Hence, the block diagram of this module is identical to Figure 3.5. The main difference with the rest of the probes is located in the *Test Block*. This block has been divided into five states, corresponding to the automated OSSTMM phases: Visibility assessment, Accessibility testing, Circumvention research, Access cracking and Survivability testing, also called Denial of Service state. The flow diagram of the *Test block* is shown in Figure 3.8 and the functions of every state are the following:

- **Visibility assessment**: This phase performs an inventory of the wireless devices on the detected networks. The configuration file sent by the *Management Module* specifies the scope and the time that the probe will spend capturing traffic, due to the fact that the probe will not be associated with any specific wireless network during that time. A file for every detected network will be created, including every detected AP previously sent in the configuration file (*In_scope*), potential rogue APs that have been detected but were not included in the configuration file (*Out_scope*) and, finally, APs not detected but included in the initial configuration file (*No_scope*) (see Figure 3.8). The last ones do not need to be tested, as they appear to be off-line.

- **Accessibility testing**: This phase gathers information about how easy is the access to the detected WLANs. It provides data about the used channel, encryption, authentication, IP range, SSID and DHCP. IEEE 802.11 standard uses WEP for data confidentiality but it has been proved to be ineffective. The encryption protocol used in WEP has been severely compromised [45] and WEP-cracking software is widely available off the Internet. If the WEP keys have been included in the configuration file, the probe will try to decrypt captured frames and results will be stored in the *Results1* file. If the key has not been given, the system will try to obtain the key length and enough frames to compromise the key in the Access cracking phase. This information will be stored in the *Unknown WEP key networks* file.

- **Circumvention research**: This phase enumerates all unauthorized devices, including the equipments in the *Out_scope* file. It also checks MAC spoofing. *Results2* file includes rogue APs and devices with unregulated IEEE MAC addresses. These MAC addresses may be forged addresses randomly generated by intruders [77]. It must be considered that the wireless probe may detect external networks from surrounding buildings whose coverage area extends for the tested corporation. In this case, they will be considered unauthorized networks.

- **Access cracking**: This phase tries to discover WEP keys. It will need many frames to compromise these keys, already stored in the *Unknown WEP key networks* file. *Results3* file will include decrypted keys with data of the related APs and the time spent to decode them.

- **Denial of Service**: This phase performs DoS attacks when they are explicitly required. DoS objectives must be carefully planned as they can affect the whole network. This is an optional phase and it returns the device response to DoS attacks.

The testing tools are different in the wireless module, due to the fact that the wireless probe includes specific tools to test devices of IEEE 802.11 technology, specially regarding layer 1 and 2 of the OSI architecture. On the other hand, the applied methodology is the same for all the probes, the scripts generated to manage the wireless tools follow the same pattern than the rest of the probes and testing results are also sent to the *Management Module*. The integrated tools in the *Wireless Module* are enumerated in Table 3.7. This table shows the name of every tool, main remarks about every tool and the data obtained when executing every utility. All these tools have to comply with different prerequisites: open-source and free-distribution tools available in Linux, command-line available and no need of graphical user interface.

Acquired data	**Tool**	**Remarks**
Sniffer	*Kismet*	It captures traffic in WLANs
WEP cracker	*Aircrack*	It decrypts WEP keys. 1 million captured packets needed
DHCP data	*Dhclient*	It sends DCHP requests to learn about the DCHP server
Authentication type	No	No tools found to discover the authentication type
Unauthorized devices detection	No	Previous knowledge of the network is needed
DoS attacks	*Void11*	It does not work with certain drivers

Table 3.7: Wireless tools for testing in *Consensus Wireless Module*.

An advantage of the selected sniffer and WEP cracker is their passive behavior. This means they do not actively probe for networks but instead listen passively for wireless traffic to discover data and capture frames. Then no more traffic is added to the network and the rest of communications still maintain the same bandwidth when performing a security test.

It is compulsory to install the wireless probe in a device with a wireless network card with special capabilities. This wireless adapter must be capable of entering into RF monitoring mode [77]. After the adapter is in monitoring mode, it cannot associate itself with a wireless network. Therefore,

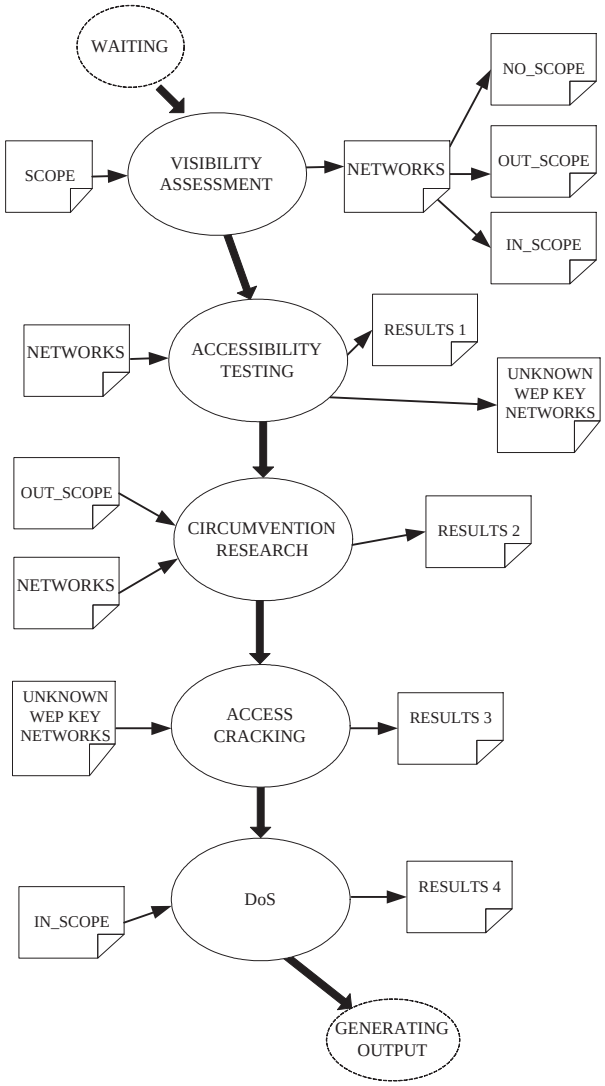


Figure 3.8: Flow diagram of the Wireless Test block.

the probe does not have access to the wireless network for other purposes. This operational mode is used to detect and sniff traffic on all the wireless networks located in its coverage area. This fact conditions the communication between the Wireless probe and the *Management module*. The adapters that comply with this requirement are listed in [25].

Once the wireless probe has finished the test passing through the different states and executing the correspondent tools, the definitive results file is generated. This file comprises all the partial files obtained in the testing states of the flow diagram (see Figure 3.8). After the wireless probe signals the *Management Module* that the security assessment has concluded, the process of acquiring the file and inserting data in the database will follow the same schedule as the rest of the probes. This process has been already exposed in Figure 3.6.

Finally, a testing module to assess security for WLANs has been presented in this section. This probe, combined with the Intranet probe, allows performing a complete security test that embraces all layers of the IP architecture and takes into account the specific characteristics of a IEEE 802.11 wireless network [25]. This contribution has been integrated in the *Consensus* framework, in order to have a security system capable of dealing with different technologies and networks.

3.6 Router and firewall vulnerability assessment

Network administrators have to be much more concerned about security rather than just protecting their servers. They also need to put safeguards in place to protect a number of networked devices, including firewalls and routers. This section discusses the requirements needed to test these devices and presents a contribution to automate this kind of security assessments. Besides, this contribution is integrated in *Consensus*, as part of the testing probes explained in previous sections.

3.6.1 Router and firewall security requirements

Routers direct and control much of the data flowing across computer networks. The border router is often the first line of defense when protecting against network attacks. Routers provide many services that can have severe security implications if improperly configured. Some of these services are enabled by default whereas other services are enabled by administrators. As another option, a firewall is a security device between two or more networks. Firewalls provide this security by filtering unused ports and opening ports to allowed hosts. Often, firewalls are employed in conjunction with filtering routers. Then overall perimeter security of a corporation benefits when the configurations of the firewall and router are complementary. Security testing provides a means of verifying that security functions are compatible with system operations and that firewalls and routers are configured in a secure manner.

A router's primary purpose is to forward packets between networks toward their destination. In addition, a router may be responsible for filtering traffic, allowing some packets to pass and rejecting others. It can also be used to partition networks and restrict access to certain networks or services. In this way, filtering is a very important responsibility for routers as it allows them to protect network devices from illegitimate traffic. When performing this task, routers would be acting as a limited type of firewall. Consequently, forwarding and filtering capabilities of routers should be tested to detect possible vulnerabilities and weaknesses in both functions.

Exposure of a router can lead to many security problems on the networks connected by that router, or even other networks with which that router communicates. As regards forwarding capabilities, compromise of a router's routing table can end in less performance, denial of communication services, and exposure of sensitive data. Moreover, compromise of a router's access control can end in exposure of network configuration or denial of service, and also can ease attacks against other network devices. Related to filtering capabilities, a weak router filtering configuration can decrease the overall security of a corporation network, expose internal devices to attacks and facilitate hackers to avoid detection. In contrast, well-configured secure routers can greatly improve the overall security of a network. As a result, a security test should also consider the analysis of network routers.

The OSSTMM methodology also reflects the necessity to test routers in the Internet Technology Security section, as detailed in Table 3.3. Security assessments should test against several services and features on the target routers in order to identify the router type and the features implemented. Information about supported routing protocols should be gathered, as well as data about access control and HTTP services. Many routers can be web-based remote administrated and this service, when improperly configured, can become an open door to the router and, afterwards, to the rest of the network. Additionally, other services should be detected and analyzed. Access control lists should be tested to verify the filtering process. This last phase must be performed when testing firewalls too.

A firewall can provide additional access control over connections and network traffic. Firewalls are usually placed at the boundary of the network to block unwanted traffic from or to external networks, as well as to regulate traffic flow between domains inside the same network [20]. Some

firewalls can check addresses and ports and look inside the packet header to verify that it is an acceptable packet.

The OSSTMM methodology includes the firewall security assessment in the Internet Technology Security Section as well. In detail, the Access Control Testing or Firewall Testing phase is in charge of this assessment, as shown in Table 3.3. This phase is designed to verify the filtering capabilities of firewalls, assuring that only what is expressly permitted will be allowed into the network, and anything else will be denied. Additionally, the tester should be able to understand the configuration of the firewall and the mapping it provides through the servers and services behind it. Then, the test should collect data identifying the firewall and its features. It also should gather information regarding which protocols and packets can traverse the firewall and which others are filtered.

3.6.2 Router and firewall tests in Consensus

Routers and firewalls are important elements in a network and many information can be obtained when testing this kind of devices. A methodology must be followed to test them as these devices are core elements in network security. On the other hand, different testing tools are needed to perform every part of this test and gather results. This is why their security assessment should be also automated. This automation is the next contribution presented in this section and it has been integrated in the *Consensus* framework as well.

The security assessment of routers and firewalls is included in the diagram flow of the wired testing modules shown in Figure 3.7. It conforms to the Router testing and Firewall testing phases. These phases will be explained in more detail in this section due to the fact that the process followed to test these devices is significantly different to other network devices.

The first step has to perform a vulnerability assessment of routers and firewalls. This step is also called *Vulnerability scanning* as it resembles the function of the Vulnerability scanning phase of a wired testing module. However, specific plugins related to routers and firewalls are used to detect distinctive vulnerabilities particular to this kind of devices. Secondly, the test is in charge of detecting all the active protocols in the device. This phase is called *Protocols detection*. Thirdly, routing tables and information regarding routing protocols and autonomous systems must be discovered. This is the *Routing information* phase. Finally, the filtering policy is checked in the *Filtering validation* phase. The flow diagram of this security test is shown in Figure 3.9.

The behavior and procedures to gather data in the Vulnerability scanning, Protocols detection and Routing information phases are identical. Different open-source tools are automated to perform every function and their results are processed at the end of the execution. Only one probe is needed to perform these phases. In contrast, the main difference to any of the other testing phases, including wired and wireless modules, is located in the Filtering validation phase. This phase has to verify the firewall's ability to block or permit certain traffic. Moreover, it has to test not only firewall outbound capabilities from the inside, but also how the firewall is egressingly filtering local network traffic.

The implementation of the Filtering validation phase requires two probes: a probe outside the network and another probe inside the network. One of these probes will send known and specific packets to pass through the firewall, whereas the other probe will sniffer these packets on the other side of the firewall. Finally, a results file is obtained in which the sent and received packets are included. These two profiles can be compared to check the filtering capabilities of that firewall. However, a previous study about the best traffic to inject into the network should be performed. For example, if the packets sent by the probe are not correlated with the filtering rules, all of them will enter into the network. In this case, the tester will know that there are no filtering rules associated with that traffic, but it will not obtain any information about other used rules. Still, the more number of packets to send to the firewall, the more possibilities to deduce the filtering rules,

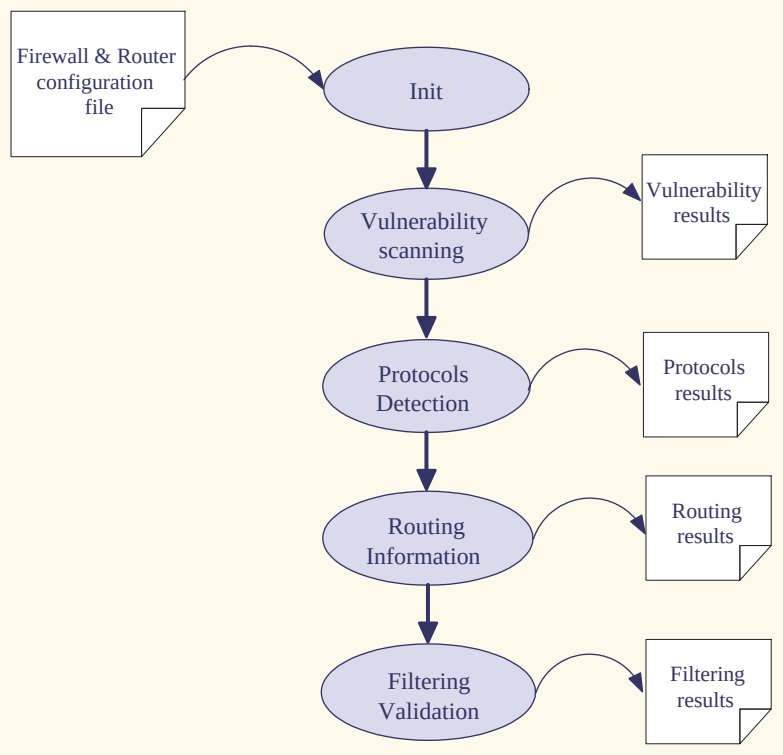


Figure 3.9: Flow diagram of the Router/Firewall Test block.

but also the more time needed to finish the test and obtain some results. Thus a foregoing analysis of the testing scenario should be done so as to reduce background traffic and testing duration.

The security assessment of routers and firewalls demands for the use of new testing tools to obtain the required information. These tools have to comply with the same prerequisites as all of the other utilities integrated in *Consensus*, as explained before: Linux open-source and free-distribution, command-line available and no need of graphical user interface. After analyzing different security tools, Table 3.8 shows the most adequate ones that have been selected to be included in this module. Table 3.8 summarizes the security testing tools chosen and the information acquired for each phase of the flow diagram of a router/firewall assessment. The execution of these tools has been automated and their results have been included in the final results file sent to the *Management module* to be stored in the system database.

Testing phase	**Tools**	**Acquired data**
Vulnerability scanning	*Nessus + router/firewall plugins*	Specific vulnerabilities of routers and firewalls
Protocols detection	*Irpas + Protos*	Active protocols in the router or firewall
Routing information	*Irpas + ASS*	Routing tables, routing protocols and autonomous systems data
Filtering validation	*Ftester*	Sent packets, received packets, dropped packets

Table 3.8: Tools for firewall and router testing in *Consensus*.

3.7 Chapter summary

Data networks are playing a more important role in today's life and many crucial services are becoming more dependent of them. As a result, security is becoming a compulsory requirement in any network, due to the fact that networks have turned out to be appealing targets for hackers. Like in daily life, the more protected a network or a building is, the less possibilities to be attacked. Then, the effort spent to penetrate a secure network is not worthy compared to the possible benefits. For that reason, the main goal of every security network administrator should be to protect their networks with the maximum level of security but, as technologies and threats evolves, also this security level should be always maintained.

Attackers tend to identify security weaknesses and holes to take malicious advantage of them. Hence, a good way to protect a network is to know in advance these weaknesses to fix the breaches before they can be taken advantage by others. Thus security assessments should be periodically performed to detect new vulnerabilities and fix them immediately. This chapter has focused on a very important phase of a security assessment: the testing phase. The main goals of this stage and the different methodologies and procedures related to the testing phase have been presented and analyzed. Also the complexity and the implications of performing a thorough security test have been exposed and several limitations have been identified. A global framework to solve these limitations has been proposed in this chapter. This framework is the *Consensus* system and it has been presented as a contribution of this thesis [25, 29, 30, 31]. *Consensus* has been described as a new platform to perform security assessments that improves the processes related to the testing phase by automating the main procedures and using an open source methodology. This large contribution has been divided into different components, in order to explain the proposals related to the different improvements with more detail. Therefore, these contributions have been focused on better performing security assessments on wired and wireless networks, taking into account not only servers and computers, but also other main network devices like firewalls and routers. The particular contributions for each of these environments have been integrated in the new system *Consensus*, providing a new valuable and practical framework for security experts.

With the use of this new framework *Consensus*, the security tester daily work has been improved, due to the fact that *Consensus* has automated the testing phase of a security assessment and it can store the results gathered in the different tests. However, all the compiled information is simply preprocessed before showing it to the security analyst. Thus the analyst work is similar to a manual test, except for the web interface that can show data collected from different security tools. Analysts must handle all the information and obtain conclusions based on the data and their own expertise. Contributions presented in Chapter 5 will improve the *Analysis module* of *Consensus*, so as to alleviate the work of security analysts by processing testing results and helping analysts in the study of the gathered data.

Chapter 4

Consensus: Experimentation

The main goal of the experimentation is to corroborate not only that the *Consensus* framework has been designed and implemented correctly but also that it fulfills security tester expectations. Experiments should then verify that *Consensus* improves the testing phase in a vulnerability assessment. This chapter describes the set of experiments that we have conducted using *Consensus* to evaluate its performance in real data networks. The different testing scenarios are detailed and experimentation results are depicted. Our evaluations demonstrate how *Consensus* provides improvement in the daily work of a security tester.

4.1 Introduction

The *Consensus* framework has been presented as a main contribution to address several issues related to the testing phase of a security assessment. A manual security test often requires the use of different testing tools to gather security data. Their output is not always compatible as data is provided in different formats. The scattering and incoherence among the outputs also difficult the analysis. In addition, security testers need specific knowledge to configure and run these tools, and manage their results afterwards. On the other hand, an automatic security test may reduce significantly human intervention. Moreover, it may lessen the expert knowledge of the different testing tools separately, focusing only on a single framework: *Consensus*. Finally, the centralized storage of the data gathered produces a single data source with consistence and lack of dispersion.

This chapter is devoted to the evaluation of the *Consensus* framework proposed in Chapter 3. We have tested *Consensus* with different real scenarios where several vulnerability assessments over different data networks have been carried out. Security tests have been executed both manually and automatically to verify the benefits of the automatic contribution: *Consensus*.

Vulnerability assessments have been run on different scenarios. First of all, *Consensus* was successfully tested on a testbed in the Networking Laboratory of the Computer Science Department of La Salle - URL. This experimentation was mainly oriented to check whether security tests may have caused disruptions to network devices or affect network performance. Automated test execution may require a significant network bandwidth. Preliminary results where also unpredictable on this field as a system like *Consensus* never existed before. The successful results of this preliminary experimentation allowed us to use *Consensus* in real operating networks. The second scenario included the data network of La Salle. This network was tested twice within a 3-year period, so different vulnerabilities were found in both assessments for the same network. Finally, *Consensus* has been used for a security test on an external company. This scenario has dealt with a data network from a corporation whose main characteristics were firstly unknown, providing a complex environment from the enterprise world. Results have shown that *Consensus* not only improves flexibility of network testing tasks but also provides a cost reduction in terms

of time. All this experimentation and results are detailed in the next sections.

4.2 Experimentation on laboratory testbed

The first phase of a security assessment is to test devices that constitute a network. This is why different security tests have been run using *Consensus* not only to check how the system works but also to obtain enough data to further analysis. Therefore, data stored in *Consensus* comes from real tests and vulnerabilities discovered in network devices.

First security assessments have been executed over a security testbed designed and implemented for this purpose. This testbed is located in the Networking Laboratory of La Salle. It contains a wide variety of devices, including servers with different OS and services, user hosts, wireless devices and even IPv6-compatible devices for future IPv6 security studies. This broad representation of devices has been feasible by the use of the virtual machine technology that minimizes the number of physical machines. The testbed topology is shown in Figure 4.1. It is composed of three wired networks: the intranet, the DMZ and the external network, all three interconnected by a firewall.

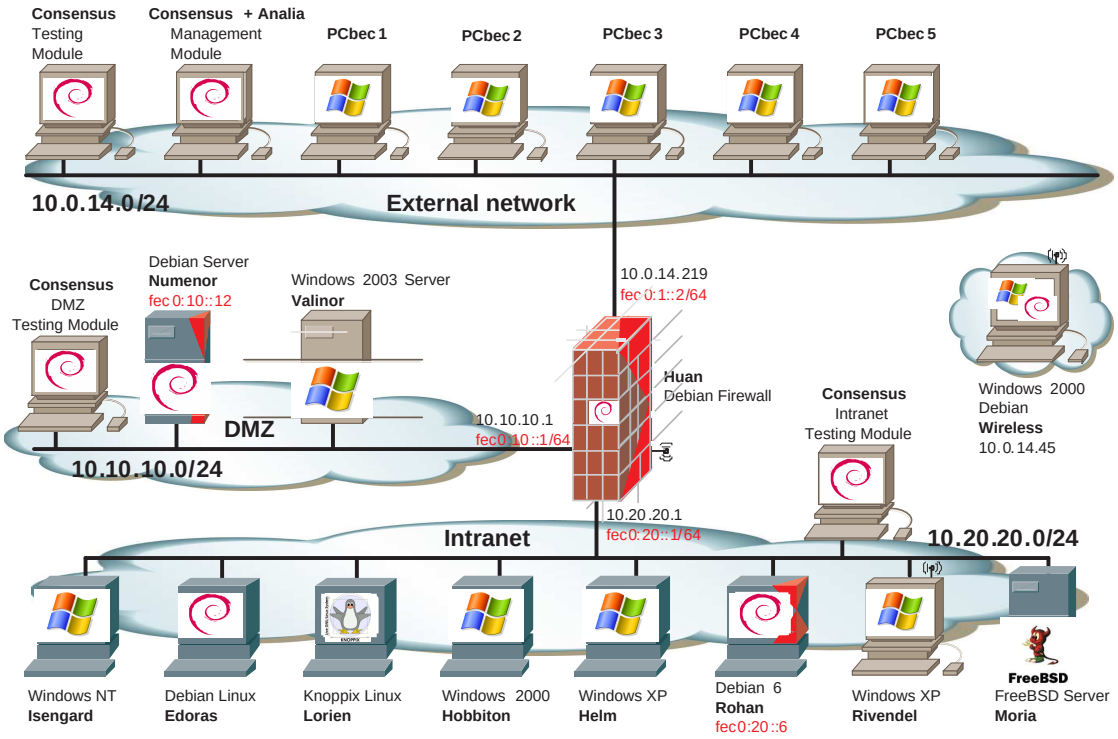


Figure 4.1: Experimentation testbed of the Networking Laboratory of La Salle.

Consensus has been set to test all the devices located in the testbed. The *Management Module* has been installed in the external network to control the whole system. The different probes have been placed separately: The *Intranet Testing Module* in the intranet, the *Internet Testing Module* in the external network, and the *DMZ Testing Module* in the DMZ network. A *Wireless Testing Module* has been also installed to test the wireless devices of the external network.

Different security tests have been programmed using *Consensus* web interface. An example of a test schedule is shown in Figure 4.2. It displays the range of IP addresses to scan, the IP address of a router to be tested and the IP address of the firewall to be tested. In the case of Figure 4.2, the IP addresses and the related devices correspond to the external network.

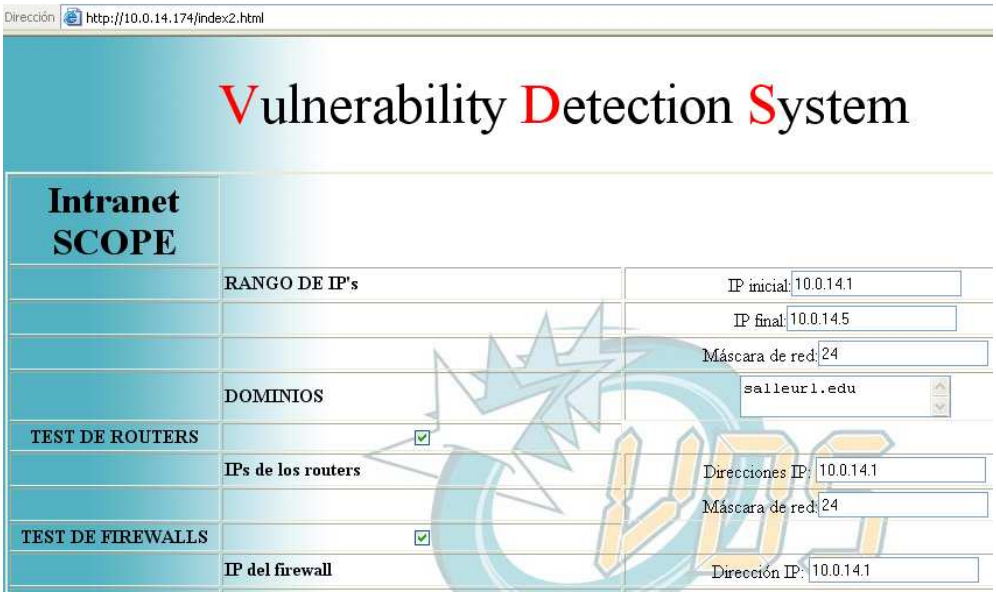


Figure 4.2: Management console of Consensus: Test configuration.

The execution of the different testing tools has been automated properly and the information has been stored correctly in the central database. *Consensus* database acts as a repository for results obtained from security testing. It also provides reports to review test results using the web interface. An example of a summary of test results is shown in Figure 4.3, whose devices also belong to the external network. This capture shows the IP addresses of the devices tested and a link to other pages to read the specific information of each device and the details about the detected services and the vulnerabilities associated to them.

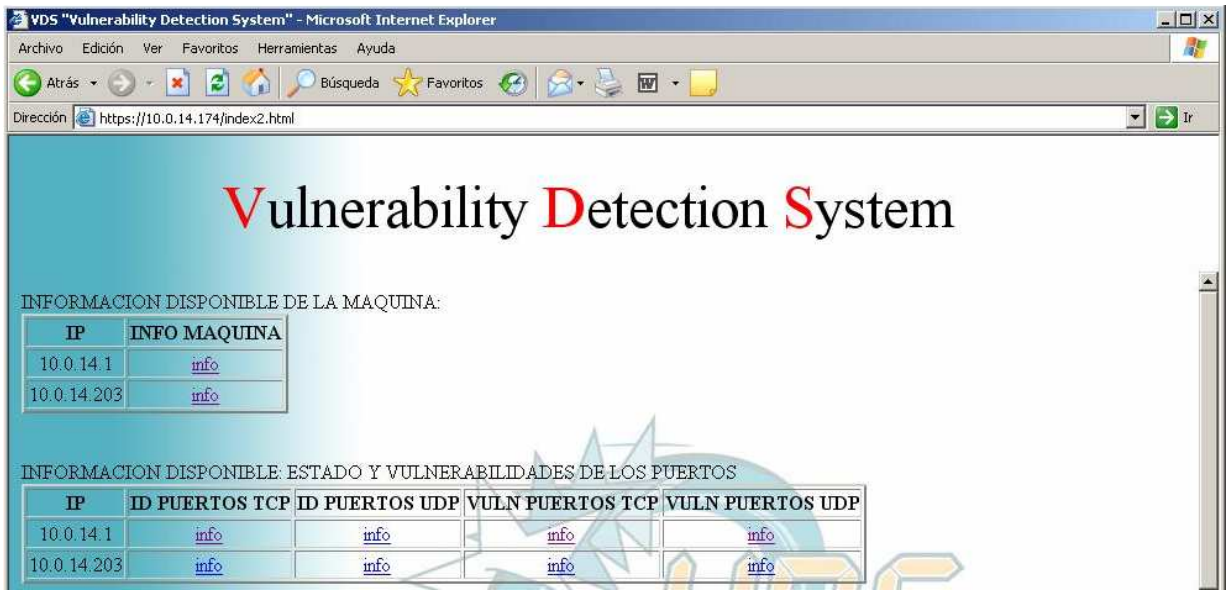


Figure 4.3: Management console of Consensus: Summary of Test results.

Router and firewall vulnerability assessments have been executed in this testbed too. The experimentation has been applied to the router that interconnects the intranet, the DMZ and the

external network. This router is shown in Figure 4.1. Besides, this device acts as a firewall by having implemented different filtering rules. Then *Consensus* has been tested to validate its use in firewall vulnerability assessments.

Additionally, wireless security tests have been run for experimentation. The management console of *Consensus* for wireless testing is different from the graphical interface for wired testing. Some forms need to be fulfilled before running any wireless test like, for example, the IP range or SSID. These are some of the parameters that the *Management Module* has to send to the *Wireless Module* before starting the test. An example of the graphical interface for wireless testing is shown in Figure 4.4. In this form, the sniffing interval where the probe will not be associated to any access point must be specified. The valid SSID and the IP address of the access point are other variables to be pointed out. If the WEP key is specified, *Consensus* will automatically decrypt the encrypted messages. If the key is not specified, *Consensus* will try to discover it by using its integrated wireless tools.

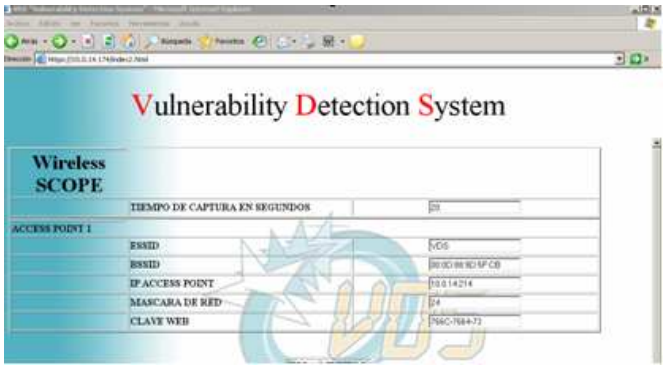


Figure 4.4: Management console of Consensus: Wireless test configuration.

The results from a wireless security test are also stored in *Consensus*. They can be processed any time after the security test. Information about the obtained ESSIDs, MAC addresses and manufacturers, channels, radio frequency, power, signal to noise ratios (SNR), IP ranges, WEP keys and other data useful for the security expert can be shown in an intuitive view. This information can help auditing a wireless network and finding possible vulnerabilities, rogue APs or unauthorized clients. An example of a summary of wireless testing results is shown in Figure 4.5.

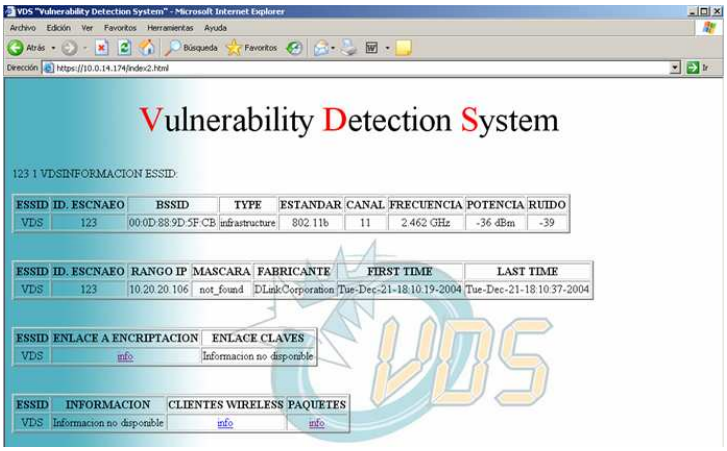


Figure 4.5: Management console of Consensus: Summary of Wireless Testing results.

The overhead in the network produced by *Consensus* has been analyzed to study possible drawbacks of the system proposed. Two different flows must be taken into account: on the one hand, the traffic flow between the *Management module* and the probe; on the other hand, the traffic flow between the probe and a tested device. These flows have been studied with the sniffer Wireshark and the ACE software (Application Characterization Environment) from OPNET. The ACE software provides effective performance management throughout the application life cycle. It describes the *Consensus* application and analyzes its communications behavior related to the existing network traffic so as to detect bottlenecks.

The traffic flow between a testing probe and a tested device is sporadic. It is only active when doing a security test. This traffic is generated by the testing tools that poll the tested device to obtain information. Therefore *Consensus* does not add overhead in this flow, due to the fact that the system generates the same frames to poll the devices that would be generated by the single testing tools when executing them manually.

The traffic flow between the *Management module* and a testing probe differentiates two stages: *Waiting* mode, where no assessment is being executed, and *Testing* mode, where the results file must travel along the network after the assessment to be stored in the central database. In *Waiting* mode, only keepalive messages and their response are sent through the network. This is a constant but minor traffic, as very small frames are periodically sent. In *Testing* mode, once the assessment has finished, the probe signals the *Management module* to inform that the results file is ready. Then, the results file must travel along the network until it arrives to the *Management module*. This is an occasional traffic that is only present when a test has ended. The bandwidth of this traffic may vary, depending on the number of tested devices whose data is sent in the same results file, as explained in Section 3.3.7. An example of a traffic flow analyzed by ACE when sending a results file to the *Management module* is shown in Figure 4.6, circled in red. This figure shows the length of the different packets and the time spent to send them. The frames sent by all these flows are encapsulated in SSH, so sniffers can only detect the number of frames and their size but not their contents.

Consequently, when using *Consensus* in an operating data network, a previous scheduling should be done to consider the amount of bandwidth required to communicate the probes and the *Management module*. This bandwidth is needed occasionally after a security test has finished, but it has to be considered as a requirement to make the system work properly. This is a plain calculus that will increase its value linearly according to the number of devices to test in the same assessment, due to the fact that the most part of the overhead will be produced by the results file. This file integrates the results of the different individual testing tools included in *Consensus*. The bandwidth needed between the probes and the tested devices should be also calculated. But this value is not influenced by *Consensus* and it should be considered even in a manual security assessment.

Consensus simplifies and enhances network security experts' work as processes related to security testing are managed from a single point. Moreover, all the information collected is stored in the system database for its future analysis. A corporate network can handle many devices, thus a thorough test can result in a great amount of data that is stored centrally when using *Consensus*. Security testers only need to know how *Consensus* works and they do not need specific knowledge about all the simple tools that *Consensus* automates. It is important to mention that *Consensus* finds the same vulnerabilities as the single tools. So the improvement is not on the number of the detected vulnerabilities but on the way they are detected, the integration of the different individual testing tools, the optimization of the knowledge needed and, the time and effort spent to detect them due to the automation of the main tasks.

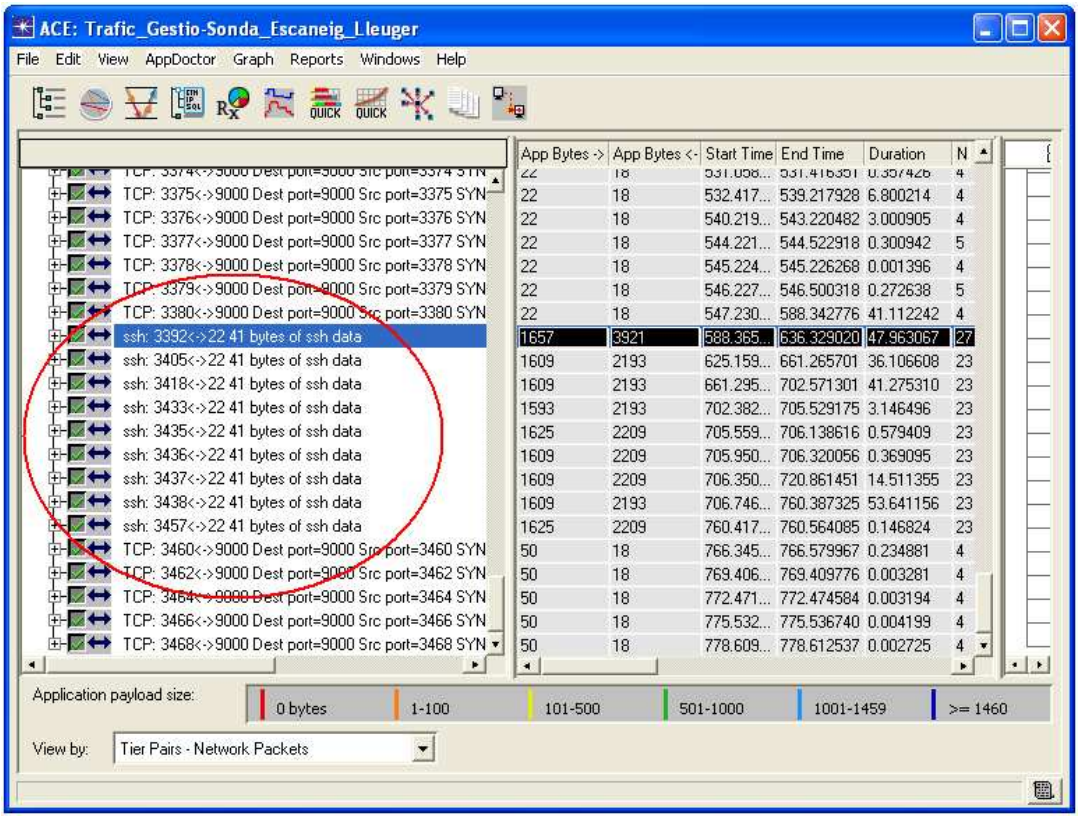


Figure 4.6: ACE: Traffic flow between Consensus management and a probe sending a results file.

4.3 Experimentation on La Salle Network

Security tests have been executed over the university network of La Salle to obtain data from real servers, student laboratories and staff computers. These security tests were authorized by the CSI, that stands for *Centre de Serveis Informàtics* (Computing Services Center) of La Salle. The data obtained was available to CSI to improve the security of the corporate network. This network interconnects different types of devices, more heterogeneous than the devices from the Networking Laboratory. So the performance of *Consensus* can be tested in a real environment, where traffic from *Consensus* has to coexist with the rest of the network traffic.

Two main security assessments have been done to collect data from the network devices of La Salle. The first test was done in April 2005 in order to check the first implementation of *Consensus*. A planning to test the different network devices was approved by the CSI, so *Consensus* was programmed to scan the allowed network devices in a scheduled timetable. In this security assessment, 44 devices were tested: 21 (14+7) from two different student labs, 9 public servers, 11 internal servers and 3 staff computers. Testing results from these devices were stored in *Consensus* database. Many different vulnerabilities were detected and reported to be solved.

A second security assessment was done in April 2008 to detect new vulnerabilities in the main network devices of La Salle. In this assessment, the individual tools from *Consensus* were upgraded in order to include the newest vulnerabilities, services, operating systems and platforms. In addition, several improvements were considered in these new versions. This second assessment allowed us to compare testing results of the same devices in different time intervals. So a comparison of their level of security was also possible. An authorization from the CSI and an accepted planning of testing schedule allowed us to do the assessment without interfering with the normal operation

of the university network. This assessment tested 90 different devices: 46 devices from four student labs (14+12+12+8), 19 interval servers, 9 external servers, 10 staff computers and a computing cluster of 6 components. All these devices provided a wide range of contrasting elements, with a variety of Windows, Linux, Sun Solaris, Mac and even a proprietary hardware with a customized operating system. Information regarding all of them and data about their open services, operating systems, general and specific vulnerabilities was gathered by *Consensus* and stored in its database. These results were also supplied to the CSI to correct the detected vulnerabilities and minimize the security holes.

The utility of these assessments and, therefore, the benefits of *Consensus* have been corroborated with the CSI. The integration of the different testing tools and their management via a unique interface included in *Consensus* allow testers to minimize the time needed to do a security test. The upgrade of *Consensus* includes the upgrade of all the individual tools, reducing also the efforts to bring up to date the different applications. The possibility of scheduling the tests gives flexibility. It allows the system to do the security tests in periods of low traffic load or when testers consider more appropriate so as not to impact on the whole network performance. The centralized storage of the results gathered allows also an improved information retrieval that can be queried at any time after the test has finished. Besides, this storage allows to compare different tests over the same network to check whether the arrangements implemented in the network to solve the detected problems are really effective or not. Finally, the execution of these assessments in a real network has allowed us to verify that *Consensus* is capable of working in a real environment without disturbing the day-to-day traffic. The compilation of all these results will be useful to do the experimentation of the analysis phase of a security test.

4.4 Experimentation on an enterprise network

A final experiment has been done to validate the performance of *Consensus*. It has been used to test the network security of an external enterprise. We have tested this network both manually and automatically with *Consensus* in order to evaluate its efficiency.

This corporation required to generate the test from outside in order to verify the possible inputs to its network. From outside only two devices were visible, so these have been the ones to be checked. IP addresses and names of devices have been changed to protect the privacy of the network' owners. Although a thorough security test includes a denial of service attack, this option was disabled on client requirements. Table 4.1 summarizes the time spent to do a manual test and the period of time spent to do an automatic test with *Consensus*.

Hosts	**Manually**	**Consensus**
10.0.0.1	2 day/man	16 hours
10.0.0.2	3 day/man	22 hours

Table 4.1: Time period of a manual and automatic security test.

What gives added-value to the automation of *Consensus* is the fact that while a manual test spends time in working hours (8 hours/day), an automated test spends both working and non-working hours. Tests duration is similar but the measurement unit makes a difference. This fact not only improves flexibility of network testing tasks but also provides a great cost reduction. It also allows security testing to be performed during non-working hours to minimize effects over networks. Moreover, manual verification of tools' success and time required to write reports dramatically delays the testing process. In contrast, the automation of Consensus saves a considerable amount of time on verification and report writing tasks. It is important to mention that both methods find the same vulnerabilities as they use the same tools. In this experimentation, no serious

vulnerability was detected; 33% of the detected vulnerabilities were regarded as high security risk and the rest were considered to be low security risk.

4.5 Chapter summary

Experiments provide a method to support *Consensus* performance and behavior. For this reason several experiments have applied the system *Consensus* in different types of networks to do controlled security tests.

First experimentation has executed security assessments on a contrived laboratory environment, the Networking Lab of La Salle. A particular testbed has been designed for this purpose. In this scenario experimental conditions have been controlled with more precision and certainty. Traffic flows between the components of *Consensus* have been modeled to detect possible bottlenecks. The system has correctly automated the processes related to an OSSTMM test and has properly gathered all the information after a security test.

Different experiments have run *Consensus* on La Salle data network. These experiments have been coordinated with the CSI so as not to collide with the usual network flows. This collaboration has allowed us to obtain useful results from *Consensus* to solve security holes in the network. These assessments have been done in different periods of time. Then, a comparison between the diverse results of the same network allow security testers to detect whether changes and upgrades have patched the network correctly.

Finally, a security test to a corporate network has been executed manually and automatically using *Consensus*. The duration of the tests has been measured. The possibility of running automatic tests with *Consensus* in non-working hours allows the system to diminish the total time spent in the test.

As stated in Chapter 3, the experimentation has validated that *Consensus* automates the testing phase of a security assessment. Additionally, test results are gathered and stored in a central database that can be queried when necessary. These statements demonstrate the main goal of *Consensus*: the improvement of a security tester's daily work. Now it is time to improve the security analyst work as well.

Chapter 5

Analia: Improving Analysis and Reporting phase in a vulnerability assessment

Previous chapters have proved that a security assessment is an effective mechanism to identify vulnerabilities and weaknesses in systems or networks before they are exploited. However a manual analysis of security assessment results is an extensive time consuming process and demands for large expertise. For that reason, network security testing is currently requiring new ways of finding patterns in large data sets in order to analyze information obtained after a security assessment. This chapter details the contributions of this thesis to the analysis phase of a vulnerability assessment. Contributions to the final reporting phase are also described. These contributions have been turned up in *Analia*, the analysis module of *Consensus* that includes artificial intelligence capabilities to handle the results previously gathered in the testing phase.

5.1 Introduction

A security assessment of data networks is a basic labor that must be periodically executed to maintain a proper level of security. The last phase of a security audit implies a structured analysis of the data collected and a reporting approach [97]. Nevertheless, a comprehensive network security analysis must coordinate diverse sources of information to support large scale visualization and intelligent response [34]. In addition, logs collection, network traffic capture, vulnerability fingerprinting, intrusion detection, and potential threat identification are tasks difficult to handle when managing large data sets [42, 81]. Consequently, there is a need for an integrated system that automates the execution of security tests, gathers all the information obtained in every step of the test, processes all this information and helps discovering errors, misconfigurations and vulnerabilities of a data network.

The wide variety of technologies, systems and devices, together with the increase of vulnerabilities and threats, force security auditors to work with multiple testing tools and large amount of data results. These results are obtained not only from the security assessments but also from the different security devices collocated in the network. Moreover, a corporate network can be composed of many network devices, thus a thorough test can result in a great amount of data. Therefore, trying to manually find a behavior pattern or certain vulnerabilities becomes an arduous task. Security applications require some intelligence to recognize malicious data and unauthorized traffic. These applications should identify intrusion and vulnerability data patterns, learn from previous decisions and also provide a proactive security policy implementation without having to analyze all the gathered data. So an automation of the different processes included in a security assessment becomes essential.

The topic of interest related to the automation and compilation of information after a security test has already been solved with the different contributions presented in Chapter 3, consolidated in the *Consensus* framework. The next contribution of this thesis is focused on the management of security assessment results, in order to improve the deduction of conclusions regarding the security level of a network. This line of research is based on the design and implementation of a system capable of automatically handle and analyze the data obtained after a security test. This new approach will ease the tasks of security analysts, by assisting in the processing of the data from detected vulnerabilities and security threats in a network and its devices. This contribution will be applied not only on wired but also on wireless environments and extensible to other technologies due to its modular design.

This thesis contributes with different proposals to improve the analysis phase of a security assessment [28, 26, 47, 23, 24, 27]. All the contributions of this thesis related to this phase are integrated in the *Analysis module* of the *Consensus* framework. This module was in charge of showing the security test results that the different testing probes had gathered, as explained in Section 3.3. The improved analysis module has been named *Analia* and it includes several artificial intelligence techniques to enrich the management of testing results. Its design, architecture and implementation will be explained in this chapter. This research work has been partially supported by the Spanish grant CIT 390000-2005-27 from the *Ministerio de Ciencia y Tecnología* as a part of the project *Analia - Sistema de Análisis de Detección de Vulnerabilidades mediante Inteligencia Artificial*, and also with the project TIN2006-15140-C03-03 (*MID-CBR: Un Marco Integrador para el Desarrollo de Sistemas de Razonamiento Basado en Casos*).

This chapter describes all the contributions related to the last step of a security assessment, the analysis phase. First of all, the application of artificial intelligence and machine learning methods in this data security domain are studied. Special attention is focused on unsupervised learning techniques and clustering approaches, due to their suitability for *Consensus* security data. Then, different clustering algorithms and cluster validation approaches are analyzed in detail. The most suitable solutions are selected to be included in *Analia*, the improved proposal of *Analysis Module* of *Consensus*. Afterwards, the global architecture of *Analia* is expounded and the equivalence among its different modules and the clustering process steps is specified. As a result, *Analia* is presented as the global contribution for the analysis of data provided from security tests. *Analia* agglutinates the different contributions referred to each of the phases of a clustering process. So different proposals for the representation of knowledge contained in *Consensus* regarding tested devices are discussed in this chapter. The design, implementation and inclusion of these proposals in *Analia* are also described. The incorporation of the studied clustering algorithms in *Analia* and the management of their results is defined too. The process of cluster validation is another step where several contributions are described. The most relevant cluster validation indices are included in *Analia* and their use and utility is shown. In addition, new validation indices *ad hoc* to this security domain are proposed. They are named *Cohesion factors*. The usefulness of all these indices to predict the number of clusters in *Analia* is also depicted. Besides, new proposals for explaining clustering results are set forth. First, the *Anti-unification* algorithm is adapted to *Analia* domain and thereafter a new proposal, called *Detailed Anti-unification* algorithm, is detailed. The integration of both algorithms in *Analia* is also explained. Finally, a conclusions section summarizes all this chapter and enumerates the different contributions that this thesis supplies in the analysis of security assessment data.

5.2 Artificial Intelligence and clustering

This section discusses the necessity of applying AI techniques to the different phases of the analysis of the results of a security assessment. The use of AI will help security analysts to extract implicit, previously unknown and potentially useful information from data of security tests. A discussion regarding the most appropriate AI methods for this security context will be expounded. They will be selected for their inclusion in the *Analia* system, the global contribution presented in this chapter.

5.2.1 Discussion of Artificial Intelligence and Machine Learning

Security assessments can generate massive amounts of data. This stressful growth in data generates the necessity for new techniques and tools that can intelligently and automatically transform the data into useful information and knowledge. Security experts are becoming more interested in using Machine Learning techniques as an assistance to analyze results obtained from security assessments [15, 80, 81, 91, 120]. So there is a need for techniques that help identifying abnormal data from everyday network activity.

An appropriate environment for Machine Learning consists in a domain where knowledge is incomplete and not available explicitly. An example of this kind of environment is the security testing domain. When carrying out a security assessment on a network, there is no need to know *a priori* which results will be obtained. Moreover, from the same testing results in two different networks different security conclusions and remarks can be obtained. Another important characteristic of this domain is that testing results may contain uncertain, incomplete or approximate knowledge. Testing results compile the data obtained from different sources of information and these sources may differ in their outcomes. For example, when testing a device to fingerprint which services are open, some tools may detect a certain port as open, whereas other tools may not detect any response regarding that port and, therefore, they will not consider that the corresponding port is open.

Machine Learning can be classified into supervised and unsupervised learning, as detailed in Section 2.3.1. Regarding the security testing domain, which is the focus of this thesis, unsupervised learning is the technique that better adapts to this domain. A security test is performed to discover network and devices status, so no previous knowledge of the network behavior is required. Therefore, unsupervised learning can be used to automatically extract knowledge and relationships from the large piles of gathered data. In this line of work, different unsupervised learning techniques have been analyzed in order to find out the best one adapted to the domain of study. As a conclusion, clustering has been selected as the most effective technique. Clustering is the intelligent partitioning of a collection of entities. Appropriate clustering reveals the underlying structure of the given objects and, hence, clustering can be viewed as a form of knowledge acquisition [86]. Finally, clustering has been integrated in the Analysis Module of *Consensus*, *Analia*, to handle results from security tests, due to the fact that it does not need previous knowledge of data and it is able to find relationships between different tested devices.

The clustering process involves three mandatory phases: pattern representation, definition of a pattern proximity measure, and the clustering algorithm; and two optional phases: data abstraction and cluster validation, as detailed in Section 2.3.2. These steps have been analyzed, designed and implemented in *Analia* in order to cluster data information obtained after a network security test and improve data management. *Analia* will cluster tested devices in groups of devices with similar vulnerabilities. First, the clustering alternatives, the data abstraction and different existent validity indexes are explained and discussed. Afterwards, the best options are selected and thereafter, Section 5.3 details the global design of *Analia* module. Then, every step of the clustering process is explained in the system proposed with the most suitable algorithms. Section

5.4 describes the pattern representation phase. Section 5.5 explains the clustering design and the implementation in *Analia* including the most representative pattern proximity measures. Section 5.6 details the cluster validation process in *Analia* and, finally, Section 5.7 shows the contributions presented in the data abstraction phase. In conclusion, a complete clustering process is included in *Analia* to handle results of network security assessments and all its phases have been studied to improve results and adapt them to this application domain.

Next sections describe different proposals to implement clustering algorithms. The first studied clustering techniques are K-means [60], X-means [96] and Autoclass [21]. They have been selected as they are the most representative techniques of the different clustering approaches (see Section 2.3.2) and, moreover, they are the ones that better conform to the features of our domain or that can be more easily adapted. Also *Soft Computing* has been analyzed, as this approach is very convenient when dealing with many and complex data, which may contain imprecise and approximated knowledge, like this security domain. Self-Organizing Maps (SOM) [72] is a widely used soft-computing technique, whose performance in this domain has been also studied and finally proposed as a contribution to process data obtained from security assessments. MOCK [59] has been studied to be considered as another valid option. This multiobjetive approach allows analysts to obtain the best clustering solution considering different criteria simultaneously. Several cluster validity techniques are detailed an analyzed in next sections as well. Finally, the importance of explaining clustering results is pointed out and different proposals are analyzed in this chapter.

5.2.2 Discussion of clustering approaches

In this section a discussion regarding the clustering approaches that best conform to the security testing data of *Consensus* is described. The selected techniques are K-means [60], X-means [96], Autoclass [21], SOM [72] and MOCK [59]. They have been detailed in Section 2.3.2. These approaches have been successfully employed by the GRSI in other domains, so their application has been also evaluated for the security testing domain. In fact, the multiobjective approach that has been considered is CAOS [48], an improvement of MOCK designed and implemented within the GRSI.

In the security testing domain, many information regarding the state of different devices and the network is obtained. Knowledge may be incomplete and not available explicitly. Then, Machine Learning techniques like clustering can be used to automatically extract knowledge, which is a difficult and challenging problem. The main goal of clustering is to group elements with similar attribute values into the same class, considering that data is initially unlabeled and need to be classified.

Algorithms from different unsupervised learning approaches have been previously described in Section 2.3.2. Their main characteristics have been detailed to analyze their behavior when applied to the security data set of *Consensus*. All these clustering approaches are considered partition methods. Two center-based methods have been studied: K-means and X-means. A search-based method has been selected: CAOS. A model-based method based on probabilistic and mathematical models like Autoclass has been also studied. Finally a soft computing technique based on neural networks, SOM, has been analyzed as well.

K-means algorithm is one of the most popular clustering solutions due to its low complexity and easiness of application. It has been already used to solve different problems of data networks and security [15, 42, 69, 76, 80, 120]. Regarding the data set of *Consensus*, K-means conforms to the type of attributes of this problem and can be easily adapted to the system. This is why K-means was selected as the first algorithm to be included in *Analia*, in order to evaluate the viability and convenience of including an unsupervised learning technique in the system [28].

But K-means suffers three main shortcomings: the number of clusters K has to be provided

by the user, it scales poorly with respect to the time it takes to complete each iteration, and the search is prone to local minima when running with a fixed value of K [96]. The first disadvantage considerably affects *Analia*. This unsupervised domain does not have previously defined how many different groups of tested devices will be found in a network. Then several executions with different K values need to be performed to choose the best one. All these three shortcomings have been solved in the X-means approach.

X-means provides a solution to automatically estimate the K value and enhance computation time. It needs an input range of possible values of K and the algorithm outputs the solution with the best number of clusters. This solution fits better to *Consensus* domain: there is no need to previously know how many groups of devices with different characteristics exist in the tested network. Computation time is improved by performing local K-means executions, which is another advantage. Also the fact of dynamically altering K lets the algorithm empirically find better local optima. All these reasons have induced us to include the X-means algorithm in *Analia*.

The original algorithm X-means has been modified to obtain a new algorithm that adapts better to this data domain. This new approach is called *X-melan*. In this approach, missing values are supported by refining the distance function. *X-melan* has been implemented by the GRSI group. Although the mentioned shortcomings have been already solved, K-means and X-means still fail in another critical point. An incorrect selection of the initial distribution of the centroids can produce an unfavorable result. Different ways to refine the selection of the starting centers through repeated subsampling and smoothing can be performed.

Regarding model-based methods, Autoclass has been analyzed. This technique can be configured to calculate the number of clusters automatically, which is an advantage that shares with X-means. The classes are described probabilistically, so that an object can have partial membership in the different classes, and the class definitions can overlap. This fact may be useful in the security domain, as the same device may belong to different clusters because it shares some security characteristics with a group of devices and other features with another cluster. Autoclass can manage nominal and numerical data, but not ordinal data. This constraint must be considered when selecting the features that will identify input data in *Consensus*. Another restriction of this approach is that data is limited for which instances can be represented as ordered vectors of attribute values. However, *Consensus* domain instances will be represented as vectors so this constraint does not impair the system. Finally, *Autoclass* has been selected to be integrated also in *Analia*, in order to compare its behavior to the rest of the unsupervised algorithms.

With respect to neural networks, SOM has been selected to be included in *Analia* due to its soft computing capabilities. Previous experiments have demonstrated that SOM generates better results than the other implemented algorithms, not only qualitatively but also quantitatively [47]. Different proposals that use SOM as clustering method in network and security domain have been reported in the literature [36, 37, 101, 118, 119]. This unsupervised learning connexionist approach follows the same philosophy of the partition approaches, while preserving its soft computing advantages. These properties can be very useful in the analysis of the information obtained from a security test. The data gathered from different testing sources may not be exactly coincident, so incongruences and inconsistences in data may be detected. Moreover the high amount of data obtained after a security test, its complexity and uncertainty justify the incorporation of SOM in *Analia*. Another SOM feature states that, although one class has few examples they are not lost. This feature may be very convenient for the analyzed data set. For example, a small number of tested devices may have a very specific vulnerability and these devices should not be grouped in a cluster with other elements as this information would then become unnoticeable. All these facts make SOM a suitable option to be included in *Analia* to manage information obtained from security assessments. In addition, possible weaknesses related to the order of the training samples or to a possible overlearning process will be considered when integrating SOM in *Analia*.

K-means, *X*-means, Autoclass and SOM are all single-objective clustering approaches. This fact implies that all these clustering techniques guide the discovery process with a single criterion. For example, *K*-means minimizes the total within-cluster variance and tends to find spherical clusters [61]. Nevertheless, we are interested in obtaining clusterings that satisfy different criteria. For this purpose, several authors have proposed to run different clustering techniques to obtain different solutions, and then, involve the network expert into the process to select manually the best solution according to certain validation methods. With the use of a multiobjective approach, this limitation is overcome. Among the different techniques for multiobjective optimization such as simulated annealing or ant colony optimization, evolutionary algorithms have been selected since they (1) employ a population based-search, evolving a set of optimal trade-offs among objectives, (2) use a flexible knowledge representation that can be easily adapted to the type of data of this domain, and (3) are able to optimize different objectives without assuming any underlying structure of the objective functions.

Finally, the five analyzed algorithms have been selected and included in *Analia* due to their benefits to extract knowledge from *Consensus* dataset. The design of this module, the tuning of their different parameters, the management of input data for each of them and the process of their output results is explained in Section 5.5.

5.2.3 Discussion of cluster validation approaches

Clustering is an automated classification of data. However it is essential to make sure that resulting classification is actually correct and sensible. There are two possible solutions in this case: expert validation and validity indices. Expert validation is rather time consuming for an expert and hence the main benefit of automated classification is lost. So usually validity indices are used for clustering validation. Then, cluster validation refers to procedures that evaluate the clustering results in a quantitative and objective function.

Clustering validity methods can be classified in external, internal or relative criteria, as explained in Section 2.3.4. The basis of the external and internal criteria methods is statistical testing. Thus, the major drawback of techniques based on these criteria is their high computational requirements. The use of statistical tests also implies accepting the hypothesis that input data is randomly distributed. On the other hand, relative criteria does not involve statistical tests and hence demands for less resources.

The main goal of cluster validation in *Analia* is to assess that the obtained clusters contain devices with similar security vulnerabilities consuming as less resources as possible. Therefore the relative criteria approach has been selected as the one that best fits in the *Analia* domain.

Many different indices of cluster validity have been proposed in the literature for the relative criteria approach. The most significant proposals have been detailed in Section 2.3.4, such as the *C-Index* [63], the *Davies-Boudin* index [33], the *Dunn* index [40] and the *Silhouette* [103] index.

C-Index [63] is the simplest index amongst all the studied validity indexes. However, *C-Index* only measures the cohesion of each individual cluster and does not give information regarding the separation of the different obtained clusters. Both *DB* [33] and *Dunn* [40] indices evaluate the clustering structure by considering the homogeneity of each cluster and the separation between clusters. Both indices aim at identifying sets of clusters that are compact and well separated. *DB* index calculates the average distance between each cluster and its most similar one. It is desirable for the clusters to have the minimum possible similarity to each other, so small index values correspond to good clusters. On the other hand, the main goal of *Dunn* index is to maximize intercluster distances whilst minimizing intracluster distances. Thus, large values of *Dunn*'s validity index correspond to good cluster partitions. The main drawback with direct implementation of *Dunn*'s index is computational since calculating becomes computationally very expensive

as the number of clusters and data elements increase. It is also very sensitive to the presence of noise in datasets, since these are likely to increase the values of the diameter measurement and, consequently, negatively influence in the *Dunn* index calculation [56]. Finally, the *Silhouette* index assigns to each data sample a quality measure, known as the *Silhouette width* [103]. This value is a confidence indicator on the membership of that sample in a certain cluster. Afterwards, more computation is needed to calculate the heterogeneity and isolation properties of each cluster and, after all, more computation is needed to calculate the global validity index for that partition. Finally, *C-index* has been discarded as it does not give information regarding the quality of a whole partition. The other indices have been included in *Analia* to validate clustering results.

All the referred clustering validation approaches evaluate clustering results considering only the same features used to previously classify the data elements. This value is useful to validate results from Machine Learning point of view, as they search for clusters whose members have a high degree of similarity as well as clusters are well separated. However, this validation is not so useful in the network security domain. Security experts look forward to a system that validates that clusters contain devices with similar vulnerabilities. But features directly related to vulnerabilities have not been used to cluster, as their data formatting is not suitable for input parameters of the clustering algorithms. On the other hand, pattern representation of network devices mainly uses information regarding port scanning and operating system fingerprinting, due to the fact that data obtained from these two processes is what a security expert would first analyze to find heterogeneities in tested devices. In fact, devices with similar open ports and operating systems may share the same security vulnerabilities, so handling this information is critical. Also these features conform with input parameter types of the selected clustering algorithms.

As a result, the main goal of cluster validation in this security domain is to evaluate that the clustering results obtained with features related to OS and open ports, also group network devices with similar vulnerabilities. Therefore, new validation indices are proposed in this thesis as a contribution to achieve this objective. These new indices will validate that obtained clusters group devices with similar open ports, OS and also similar vulnerabilities. So these new indices will use information not only about the input data of clustering algorithms, but also about other information already stored in *Consensus* and not previously processed for clustering. This complex information is difficult to parametrize as input features for clustering algorithms, but will be used to check that the obtained clusters are compact and well separated, indicating that different clusters contain devices with different security vulnerabilities or flaws. These new validity indexes are called *Cohesion factors*. They correspond to two different factors: *Intracohesion index* and *Intercohesion index*. The *Intercohesion index* evaluates the compactness of each of the obtained clusters. This factor has to evaluate how similar are the elements within a cluster, so it has to assess that elements in the same cluster share similar vulnerabilities. On the other hand, the *Intracohesion index* evaluates the separation between clusters, so its function is to check that devices in different clusters do not have vulnerabilities in common. All of them are detailed in Section 5.6.

There exist also supervised methods of validation like, for example, *N-fold Validation* or *N-Cross Validation*, widely used to evaluate Machine Learning models [51]. However, these methods are discarded in *Consensus*, due to the fact that it is necessary to know *a priori* certain and explicit knowledge about the data set. A previous classification of the data set should be needed to know what is a correct classification or not. But *Consensus* is a complete unsupervised domain, as never before has been performed this kind of classification to a data set obtained from a security test. Thus, the obtained samples can not be previously labeled as members of a certain class. This lack of knowledge about the domain impels the proposed contribution to apply unsupervised methods to validate clustering results, like the cluster validity indexes previously explained.

5.3 Analia architecture

This section describes the design, architecture and implementation of the second main contribution of this thesis: *Analia*. *Analia* is the Analysis Module of the *Consensus* framework that includes artificial intelligence capabilities. This section also expounds the different contributions that *Analia* brings in the analysis phase of a vulnerability assessment [28, 26, 23, 24, 27, 47].

5.3.1 Global architecture of *Analia*

Analia is the Analysis Module of *Consensus* that introduces artificial intelligence techniques to improve the results analysis after a network security test. Consensus automates the security testing procedures following the OSSTMM methodology [67] in order to reliably verify network security and detect existent network or system vulnerabilities. *Consensus* includes different interactive modules sharing a similar design base to provide security operations for vulnerability detection in a network.

Consensus system architecture and its relationship with *Analia* is shown in Figure 5.1. It can be seen that *Analia* replaces the *Analysis Module* of *Consensus*, displayed in Figure 3.1. This new module is still accessed from the *Management Module* using a web graphical user interface. It can also communicate with the *Consensus* database, so as to query information from security tests or store conclusions after the application of AI techniques to analyze testing results. Not only security testers, but also analysts will get benefit of this new module, due to the fact that this new web interface will be integrated in the existent management system. Furthermore, this new framework will be more profitable for security analysts to ease their daily work.

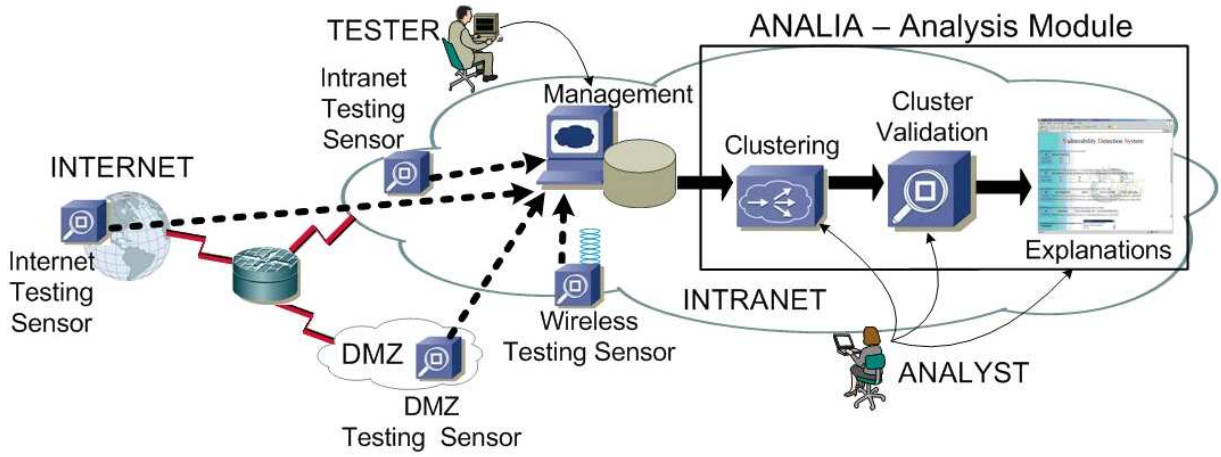


Figure 5.1: System architecture of Consensus with Analia.

The main goal of *Analia* is to help finding hidden patterns in tested devices. When a security test is done, a large quantity of devices are tested in a network. Therefore a huge volume of data is gathered. Analysts must focus on every device with the purpose of finding abnormal behaviors, incorrect configurations or critical vulnerabilities. If the list of devices is extensive some behaviors could become unnoticeable, some patterns could be masked or maybe the most vulnerable devices could be the last to be checked. In this sense, *Analia* aids the analyst to find similarities and dissimilarities within information resulting from security tests in *Consensus*. Unsupervised learning techniques can help security analysts extracting several conclusions from testing data without having to analyze the whole data set.

The process of a security test and its further analysis in a network using the system *Analia* will conform to the following phases, which are shown in Figure 5.2:

- Security test configuration using *Consensus* web interface and subsequent execution.
- Results parser and storage in the central database.
- Application of clustering techniques to data from database: samples are tested devices.
- Clustering results presentation: groups of tested devices with similar vulnerabilities.
- Validation of clustering results when required.
- Explanation of clustering results when required.

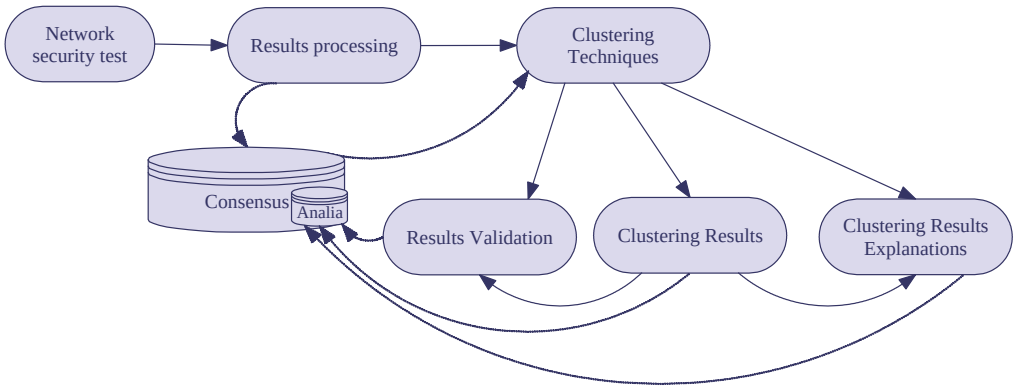


Figure 5.2: Analia phases.

The core of *Analia* is the *AI Module*, whose design will be further explained in the next section.

5.3.2 AI Module

The *AI Module* of *Analia* takes charge of the different unsupervised learning techniques and the rest of the procedures related to the clustering process. Its modularity eases the inclusion of new clustering algorithms and different knowledge representations.

The process followed by the AI module (see Figure 5.3) is summarized in the following steps:

1. The web interface of *Analia* processes the request. The clustering algorithm, knowledge representation, configuration parameters, and the devices to be analyzed can be configured.
2. The request to the AI Control block includes the parameters needed to select the specific queries that will be applied to the database.
3. The AI Control block processes the query results and generates the input and control files for the selected clustering algorithm.
4. The AI Control block calls the selected clustering algorithm and controls its execution.
5. The AI Control block obtains the clustering results.
6. The AI Control block saves details about the obtained clusters and the execution.

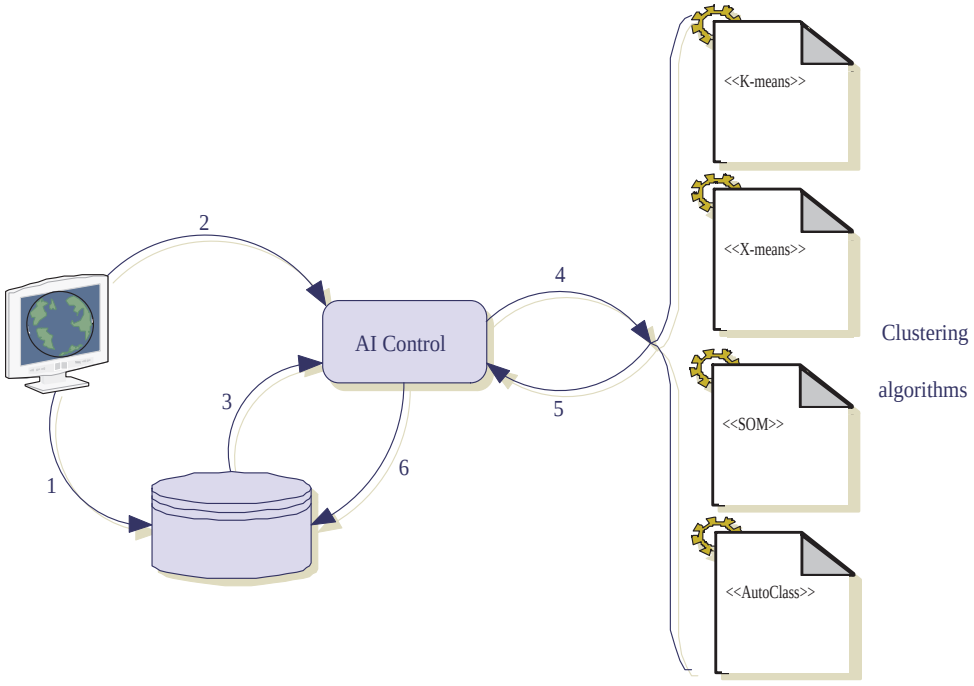


Figure 5.3: AI Module of Analia.

In addition the *AI Control block* takes over the cluster validation process and incorporates different metrics to check the effectiveness of the obtained clustering solutions. Typical cluster validation indexes calculate cluster validity by using the same features included in the data set. The most important indexes have been programmed and included in *Analia*. New indexes designed *ad hoc* to this specific domain have been also implemented in *Analia* [26]. These new indexes are called *Cohesion factors*. They validate cluster results using not only existent classifying features but also new features already stored in the database. All these indexes are useful to compare the different clustering solutions. This component of the *AI Control block* is detailed in Section 5.6.

Besides, the *AI Control block* implements the data abstraction process, where explanations that describe the common characteristics of the elements of a cluster are obtained. These descriptions use the same vocabulary of feature characterization, giving a more comprehensive justification of clustering results to security analysts [24, 27]. This feature is described in Section 5.7.

This modular architecture offers different advantages. First of all, there is a clear separation between data collection and analysis. Network tests can be regularly planned and data is stored after every test. Whenever necessary, an analysis of the tested devices can be done. Another important fact is that *Analia* is a complementary module. So analysts can choose a basic report showing the whole data set or they can process data using the AI module to refine results and obtain a more elaborated report. *Analia* works with the same database where *Consensus* stores all data obtained from the different testing tools. Thus, this central management avoids data duplication. Another feature is the knowledge representation flexibility, due to the fact that different representations can be configured for the same data set in order to introduce them in the clustering algorithms. Also the incorporation of new unsupervised learning techniques can be easily implemented when considering the same input and output file formats. Input files comply with ARFF format [7], whereas output files include the obtained clusters together with their respective devices. Configuration files are obtained from the parameters selected by the analyst in the *Analia* web interface. All these advantages are more detailed in next sections.

5.4 Pattern representation in Analia

Machine Learning and the different clustering approaches detailed in Section 5.2.1 can be very effective when managing data from a network security test. These techniques can be used to identify homogeneous patterns in data objects after analyzing their most significant features.

One of the main goals of this thesis is the improvement on the analysis of data from security tests. The grouping of tested network devices with similar characteristics can be very convenient to discover inherent structures within the gathered data. Difficulties arise in this domain due to the fact that no previous classifications have been done using this kind of information, so no previous experiences can be used to corroborate the results. In this environment, unsupervised learning is the most appropriate to manage information. However the number of groups of devices to obtain is still unknown. Every network is unique and every scan may collect different data. Hence approaches that can also discover the optimal number of clusters will be preferred in this domain. Moreover, a security scan of a network with many devices may generate large amounts of data, making not affordable a manual analysis. In this sense, clustering is useful to discover behavior patterns, similarities and differences between devices of the same network. These techniques will be used to automatically extract unknown or new conclusions that laid behind all this information.

The final objective of a security test is to identify the most vulnerable devices and detect the possible security holes in a network. The detection of these devices allows security experts to discover their flaws and perform the convenient procedures to correct and eliminate these vulnerabilities. However, many information is recorded after a security test, as it has been detailed in Chapter 3. For that reason, the selection of the main features to represent data is a very important task that must be considered before applying any Machine Learning technique. All clustering algorithms need a group of samples as input, and these samples should be described with their most representative features.

In *Analia* domain, a sample stands for a tested device from a network. The resultant clusters should group devices with similar vulnerabilities. As a result, when detecting a vulnerability in a device and finding the correct solution, all the other devices grouped in the same cluster could be healed with the same procedure without having to analyze the whole data set. This automatic grouping would help security analysts as it would not be necessary to analyze each tested device one by one and, hence, the time needed to study results from all network devices would decrement. Pattern representation can influence on clustering results: a bad feature selection will bring to poor clustering results not useful to security analysts. Then, a discussion regarding the most characteristic features for pattern representation in this domain is argued in this section.

It seems obvious that if the goal is to group devices with similar vulnerabilities, the selected features should be the vulnerabilities detected in the security assessment. However, the different security tools that are executed during a security assessment do not always detect the same vulnerabilities. Moreover, there is still no standard to define or describe a vulnerability. Different proposals exist to identify vulnerabilities and exposures, like CVE [87] or OSVDB [88] identifiers, but none of them are standards. As a result, it is possible that two different tools may detect the same vulnerability but represent it differently by using textual sentences that are very difficult to parametrize to become input parameters for clustering algorithms. In addition, most of the partitional clustering techniques are applicable mainly to numerical data sets [56]. Consequently, other features have been also analyzed to select the most adequate to clustering algorithms that fulfill the requirement of grouping devices with similar vulnerabilities. Information stored in *Consensus* of a tested device contains numerical and categorical attributes gathered by security tools. It is possible to have empty attributes in some devices, as not always is feasible to extract the same information from different devices. So it is necessary to identify the most significant features present in any tested device to become part of their pattern representation.

The first selected attributes are related to port scanning and operating system fingerprinting. Port scanning looks for multiple listening ports on a device, and detects available services associated with those ports. Operating system fingerprinting remotely determines the OS of devices. In fact, data obtained from these two processes is what a security expert would analyze first to find heterogeneities in tested devices. So handling the analysis of this information is also critical. These parameters allow analysts to get clusters that group devices with similar open ports and operating systems. When clustering these devices in the same group, they should also have similar vulnerabilities as vulnerabilities are usually directly related to the OS implemented and the open services provided in a device.

The detection of an open port in a device implies the presence of a certain service available in the device. Then that device becomes a feasible vulnerable point of all flaws and attacks related to that service. OS has an effect upon vulnerability results too. For example, vulnerabilities that affect a Windows web server may not affect a Linux web server. There is an example of the detected vulnerabilities in a Linux web server in Figure 5.4. This figure shows vulnerability results of a web server tested with *Consensus*, where it can be seen its IP address, the port associated with the web service (port 80), a description of the vulnerability obtained from the integrated Nessus tool and the phase of the security assessment where this vulnerability was detected (Vulnerability research). Another example is shown in Figure 5.5, with the testing results of a Windows web server. This figure shows its IP address, the port associated with the web service, a description of the vulnerability, the risk factor associated to that vulnerability, its Common Vulnerability Scoring System (CVSS) and the testing phase related to that vulnerability. By examining the information provided in the description of the vulnerabilities in both figures, it can be seen that they are related to the type of web server: a PHP server in the Linux device and a Microsoft IIS in the Windows device. There exist also vulnerabilities that are inherent to the OS and do not depend on the available services. Consequently, information stored in *Consensus* about detected open ports and OS has been further analyzed to be included as input features for clustering algorithms.

Thereafter, when experimentation with port and OS data has shown successful results by identifying groups of similar computers, and detecting devices unexpectedly separated from other similar devices due to changes in their configurations (see Chapter 6) [28], more data already stored in *Consensus* has been studied. This data is related to information concerning the vulnerability assessment phase. Then, a pattern representation for vulnerability information has been also proposed as a contribution.

IP	NUM PUERTO TCP	NES INFO VUL	ID APPLICATION
130.206.42.4	80	The remote host is running a version of PHP 4.3 which is older or equal to 4.3.2. There is a flaw in this version which may allow a local attacker to bypass the safe mode and gain unauthorized access to files on the local system, thanks to a flaw in the function php_safe_mode_include_dir(). Solution : Upgrade to PHP 4.3.3 when it is available Risk factor : Medium BID : 8201	vulnerability_research
130.206.42.4	80	The remote host is running a version of PHP which is older than 4.3.9 or 5.0.2. The remote version of this software is affected by an unspecified file upload vulnerability which may allow an attacker to upload arbitrary files to the remote server. See also : http://viewcvs.php.net/viewcvs.cgi/php-src/NEWS.diff?r1=1.1247.2.724&r2=1.1247.2.726 Solution : Upgrade to PHP 4.3.9 or 5.0.2 when available Risk factor : Medium BID : 11190	vulnerability_research
130.206.42.4	80	The remote host is running a version of PHP which is older than 5.0.2. The remote version of this software is vulnerable to a memory disclosure vulnerability in PHP_Variables.An attacker may exploit this flaw to remotely read portions of the memory of the httpd process on the remote host. See also : http://www.php.net/ChangeLog-5.php#5.0.2 Solution : Upgrade to PHP 5.0.2 Risk factor : High BID : 11334	vulnerability_research

Figure 5.4: Example of a vulnerability detection in a Linux web server.

IP	NUM PUERTO TCP	NES INFO VUL	RISK FACTOR	CVSS	ID PLUGIN	ID APPLICATION
172.16.1.158	80	The remote host appears to be running a version of IIS which allows remote users to determine which authentication schemes are required for confidential webpages. Specifically, the following methods are enabled on the remote webserver: - IIS NTLM authentication is enabled Solution : None at this time Risk factor : Low CVE : CVE-2002-0419 BID : 4235	Low	3	11871	vulnerability_research
172.16.1.158	80	Synopsis : Some information about the remote HTTP configuration can be extracted. Description : This test gives some information about the remote HTTP protocol - the version used, whether HTTP Keep-Alive and HTTP pipelining are enabled, etc... This test is informational only and does not denote any security problem Solution : None. Risk factor : None Plugin output : Protocol version : HTTP/1.1 SSL : no Pipelining : yes Keep-Alive : no Options allowed : OPTIONS, TRACE, GET, HEAD, POST Headers : Content-Length: 1433 Content-Type: text/html Content-Location: http://pdc.salle.url.edu/iisstart.htm Last-Modified: Fri, 21 Feb 2003 16:48:30 GMT Accept-Ranges: bytes ETag: "0c3110c9d9c21:a5e" Server: Microsoft-IIS/6.0 X-Powered-By: ASP.NET Date: Wed, 12 Mar 2008 21:08:44 GMT	Medium	5	24260	vulnerability_research
172.16.1.158	80	Synopsis : A web server is running on the remote host. Description : This plugin attempts to determine the type and the version of the remote web server. Risk factor : None Plugin output : The remote web server type is : Microsoft-IIS/6.0	None	0	10107	vulnerability_research

Figure 5.5: Example of a vulnerability detection in a Windows web server.

After selecting the main contents to be classified by the unsupervised learning approaches, the format and type of the input features must be selected. Irrelevant attributes make chances of a successful clustering useless, because they negatively affect proximity measures and eliminate clustering tendency. So a data analysis has been done to select the most adequate representations of information from port scanning and operating system fingerprinting first, and then, from vulnerability assessment.

Knowledge patterns are represented as multidimensional vectors, where each dimension is a single feature:

- Let $N = e_1, ..., e_n$ be a set of elements. An element is a tested device.
- Let $C = C_1, ..., Ck$ be a partition of N into subsets. Each subset C_i is called a cluster and C is called a clustering or partition.
- Let $e_i = f_{i1}, ..., f_{ip}$ be an element, associated with a vector of p features.
- Therefore, the feature-vector matrix has a $N \times p$ dimension.

5.4.1 Pattern representation of port scanning

The first information to be discussed is related to the detected open ports. Six different alternatives have been considered. They are the following:

1. Multidimensional vector where each feature corresponds to a port. Features are boolean and indicate whether the port is open or closed for that network device. The main problem of this representation is that a device has more than 65.000 ports, but the great majority of them are usually closed. Having a vector with so many boolean features will not provide significant clusters, as most of the features will have the same value (closed). Measuring distances between vectors with boolean values can become meaningless when vectors are too long, due to the fact that results may group devices with different open ports in the same cluster as they have in common too many closed ports.

2. Multidimensional vector where features are related only to the most commonly used ports or, for example, only the first 100 ports. Features are boolean and indicate whether the corresponding port is open or closed for that network device. The improvement of this solution is the decrease of the vector length. However, information regarding the rest of the ports is not being included in the clustering input. Then, other vulnerable ports would not be considered. Moreover, ports that are not usually open and they have been detected open in the testing phase would not be either considered and this valuable information would get lost in the analysis phase. Also the selection of the most commonly used ports is not an easy task, as they can change in every network. A very important service in a network may not be significant for other networks, as no two networks behave equally.

3. Multidimensional boolean vector where the first attributes correspond to individual ports, and the rest are related to port ranges. The value of the former indicates whether that port is open or closed. The value of the later represents whether a port has been detected open in that range of ports. This pattern representation is shown in Figure 5.6. The improvement of this proposal pursues the decrease of the vector length, and the inclusion of information related to all the tested ports. Dealing with port ranges reduces the number of features while still consideres data about all the detected open ports, information lost in option 2. The problem of this alternative is that information regarding how many ports are open in each range is lost. Furthermore, this proposal does not consider which are the ports open in the ranges. So two devices may have different open ports in the same range with different vulnerabilities. This information would be masked in this representation. Also managing only boolean features limits the pattern representation possibilities.

Ports - Port ranges --> Boolean attributes										
1	2	...	100	101 - 1000	1001 - 2000	2001 - 3000	3001 - 10000	10001 - 20000	...	... - 65000
Yes	Yes	No	Yes	Yes	No	Yes	Yes	...	...	...

Figure 5.6: Knowledge representation using ports and port ranges.

4. Multidimensional vector where the first attributes correspond to individual ports, and the rest are related to port ranges. The first features are boolean and mark whether that port is open or closed. The port range features are numerical and show how many ports are open in that port range (see Figure 5.7). This proposal improves several points. First, the vector length is decremented compared to option 1, while information regarding all the tested ports is still considered. Grouping by port ranges can be used to know whether a port is open in that range, and how many ports have been detected. So a new classification can be done taking into account the number of open ports for each range. Managing not only boolean but also numerical features can be useful to get more accurate clusters. However, information regarding which ports are exactly open in each range is still not considered. In addition, port scanning information is obtained from different automated testing tools. These tools do not always coincide with their results. Then a testing tool may have detected an open port, whereas another tool may have no information regarding the same port, considering it as closed. This information is not reflected in all these pattern presentations. Also testing tools can detect a port as open, closed or filtered. A filtered port means that a firewall, filter or some other network obstacle is blocking the port and preventing the system from determining whether it is open. But with boolean attributes, only two states can be represented: open (Yes) and closed (No). So information regarding filtered ports is lost as well.

Ports - Port ranges --> boolean/numerical attributes										
1	2	...	100	101 - 1000	1001 - 2000	2001 - 3000	3001 - 10000	10001 - 20000	...	... - 65000
Yes	Yes	No	Yes	150	0	4	1	...	...	0

Figure 5.7: Knowledge representation using ports (boolean) and port ranges (numerical).

5. Multidimensional vector where the first attributes correspond to individual ports, and the rest are related to port ranges. The first attributes are not boolean, but numerical. They indicate how many automated testing tools have detected open the correspondent port. The port range features are also numerical attributes and show how many ports are open in the correspondent port range. An example of this pattern representation is shown in Figure 5.8. This proposal includes the improvements of the previous option and incorporates the number of testing tools that confirm a port state. So a reliability measurement is reflected in the features. Nevertheless when a testing tool detects an open port, it means that the device has replied to the packets sent to that port. Consequently, it can be considered that there is a service available on that port, although other testing tools have not received any response. The techniques implemented by each testing tool may differ and they may arrive to different conclusions, but when one of the tools receives a positive response, this port can be considered open. This option still does not reflect the filtered ports detected in each device.

Ports - Port ranges --> Numerical attributes										
1	2	...	100	101 - 1000	1001 - 2000	2001 - 3000	3001 - 10000	10001 - 20000	...	... - 65000
3	1	...	0	150	0	4	1	...	...	0

Figure 5.8: Knowdlege representation using ports and port ranges, all numerical attributes.

6. Multidimensional vector where the first attributes correspond to individual ports, and the rest are related to port ranges. All the attributes are numerical. The individual port features indicate when an open port (1), a filtered port (2), or no response (0) has been detected. The port range features show how many ports are open in the correspondent port range. This proposal decrements also the vector length, while information regarding all the tested ports is still considered. As information about the tools that have detected an open port has not been considered relevant, the attributes related to individual ports reflect their possible states and, therefore, information regarding filtered ports has been included. An example of this pattern representation is shown in Figure 5.9.

Ports - Port ranges --> Numerical attributes										
1	2	...	100	101 - 1000	1001 - 2000	2001 - 3000	3001 - 10000	10001 - 20000	...	... - 65000
2	1	...	0	150	0	4	1	...	...	0

Figure 5.9: Knowdlege representation using ports and port ranges, all numerical attributes including filtered ports.

5.4.2 Pattern representation of operating system fingerprinting

Operating system fingerprinting is the science of determining the OS of a remote computer. This may be accomplished passively by sniffing network packets traveling between hosts, actively by sending carefully crafted packets to the target machine and analyzing the response, or through non-technical solution, such as social engineering. *Consensus* automates this process by using active tools that send a variety of packets to a computer, determine how it responds, and then compare these responses to a database to determine its OS. These automated tools are based in the fact that each OS responds differently to a variety of packets and the database contains information on how different OS respond to different packets.

Actually, the automated tools can provide more than one guess at the remote OS, along with a probabilistic score. Therefore, *Consensus* does not store a single result, but the different OS possibilities with their associated probabilities. Another important information is related to the version of the OS, due to the fact that the same OS may include several vulnerabilities in a version that have been solved in next versions. An example of the information stored in *Consensus* related to OS fingerprinting is shown in the table of Figure 5.10, that summarizes the device IP address, the possible OS, version and their probabilistic score. In addition, the automated tool that gathered the information is included in the table.

IP	NOMBRE	PROB	PROG
10.0.14.130	Microsoft Windows 2000 Workstation SP2	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Workstation SP3	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Workstation SP4	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Server	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Server Service Pack 1	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Server Service Pack 2	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Server Service Pack 3	70	xprobe2
10.0.14.130	Microsoft Windows 2000 Server Service Pack 4	70	xprobe2
10.0.14.130	Microsoft Windows XP	70	xprobe2

Figure 5.10: Example of data stored in *Consensus* regarding the OS fingerprinting of a device.

The pattern representation of OS fingerprinting will be a multidimensional vector that has as many features as OS have been detected in the different devices of a network. Features will be numerical and their value will indicate the probabilistic score assigned to that OS version for the represented element, the tested device. This information will be obtained from *Consensus* database, after a security test has been executed over a group of devices. An example of an element using this pattern representation is shown in Figure 5.11.

Possible OS --> Numerical attributes					
Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...
0.67	0.2	0	0	0	...

Figure 5.11: Knowledge representation using operating system fingerprinting data.

5.4.3 Pattern representation of vulnerability information

Information regarding vulnerability scanning is burdensome to parametrize due to the lack of standardization, and the different formats that automated tools return results after a vulnerability assessment. These tools return long textual sentences, like the ones shown in figures 5.4 and 5.5, that are not compatible with the format of input parameters of the clustering algorithms selected for *Analia*. This reason forced us to start working with information of port scanning and OS fingerprinting, which is easier to adapt to the input of clustering algorithms. Experimentation with this kind of data produced successful results, as it is detailed in Chapter 6. So afterwards, new features were analyzed to be added to the input samples.

Information related to the security level of the detected vulnerabilities has been also studied. The severity level can be classified in three categories: security notes, security warnings and security holes. A security note is related to miscellaneous issues. It is something to fix when getting around to it, or just some information to consider. Then, a vulnerability labeled as a security note contains information that could prove helpful to increase the security of a network, but it is not crucial. A security warning can be considered a mild flaw, like something that needs to be fixed soon. In this case, a potential situation may create a security hole if left unattended. Finally, a security hole is a severe flaw or exposure that has to be fixed as soon as possible. Each detected vulnerability is categorized by the testing tool in one of these three levels, so this information can be included easily as an input parameter to be considered by the clustering algorithms.

Knowledge representation of the severity level is symbolized by a multidimensional vector of three numerical features. Each feature corresponds to a severity level: note, warning and hole. Remind that in *Analia*, an element is a tested device and many vulnerabilities can be detected for each device. Therefore, these three features count the number of security notes, warning and holes detected in a device. Then, clustering algorithms may group devices with the most similar number of security levels by considering these attributes. An example is shown in Figure 5.12.

Vulnerability assessment tools like Nessus also classify the detected vulnerabilities regarding their risk factor. Unfortunately, risk factors are not standardized as there are no standard computer security definitions for rating risk. But they can be useful to determine the attention given to a problem. Vulnerabilities can be classified in the following risk factors: none, low, medium, high and critical. Obviously, vulnerabilities labeled as critical are the first vulnerabilities to be solved as they can generate main security problems to the tested network.

Knowledge representation of the risk factor will be a multidimensional vector of five numerical features. Each feature is related to a different risk level and counts the number of vulnerabilities with that risk level detected on a device. An example of an element with this pattern representation is shown in Figure 5.12. This representation includes also severity level features.

Severity level			Risk Factor				
Note	Warning	Hole	None	Low	Medium	High	Critical
4	3	3	4	4	3	1	2

Figure 5.12: Knowledge representation using risk factor and severity level data.

The CVSS base score [19] is related to the severity ratings for vulnerabilities stored in *Consensus*. Values from 1 through 3 receive a Low/Note rating; 4 through 6 receive a Medium/Warning rating and 7 through 9 receive a High/Hole rating. CVSS scores of 10 have a severity level of Hole but also have their Risk factor marked as Critical. Consequently, vulnerability information can be represented either by its severity level or by its risk factor, due to their direct equivalence.

5.4.4 Pattern representation for clustering in Analia

Once pattern representations for the gathered data have been analyzed, it is necessary to define the final knowledge representation for each tested device. This final representation will constitute the format of the samples file that the different clustering approaches will use as input.

First experimentations have merged information related to operating system fingerprinting and port scanning, providing a new vector to represent each device. Three different views have been selected as input files to test the validity of the selected knowledge representation. They will provide three different data sets from the same source information of *Consensus*. These three different views collect features from the same data set of a security assessment, so several comparisons can be performed regarding the different clustering tools and knowledge representations.

The three selected representations or views are the following:

1. Multidimensional vector where the features are the open/filtered ports detected in the devices of the data set and their possible operating systems. Each element or sample e_i corresponds to a device tested using the *Consensus* framework. Features related to ports illustrate when an open port (1), a filtered port (2), or no response (0) has been detected for each sample. An open port shows an available service; a filtered port means that a filter is preventing the system from determining whether it is open; no response means that there was no reply from the host. On the other hand, OS features depict the probability of having that OS installed. The value of these features is extracted from *Consensus* database. An example of this knowledge representation is shown in Figure 5.13. This representation may include more features related to ports that OS, as information of more than 65000 ports could be gathered from a single device, whereas there are not so many possible OS. This fact could unbalance the weight of the OS features for port features benefit when clustering, but it will depend on each data set. The good point is that information about the exact ports that have been detected on each device is not lost.

Open Ports					Possible OS					
23	25	53	80	...	Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...
1	2	0	1	...	0.67	0.2	0	0	0	...

Figure 5.13: Knowledge representation using all detected ports and operating systems.

2. Multidimensional vector where the first feature is the sum of the open ports of an element e_i, and the rest of the features represent the list of all possible OS with their associated probability scoring. This representation prioritizes the attributes related to OS as there is only one feature related to ports. This representation omits information about which ports are exactly open. But the number of attributes substantially decreases. An example of this pattern representation is shown in Figure 5.14.

Open ports	Possible OS					
Open ports	Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...
30	0.67	0.2	0	0	0	...

Figure 5.14: Knowledge representation using a single port range and all OS.

3. Multidimensional vector where the features regarding OS and ports are equilibrated. Then open ports are combined in ranges, and the value of the attributes related to port ranges is the sum of the open ports in every range. OS attributes still represent the list of all possible OS with their associated probability scoring. An example is shown in Figure 5.15.

Port ranges					Possible OS					
1 - 100	101 - 500	501- 12000	12001 - 20000	...	Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...
25	4	0	0	...	0.67	0.2	0	0	0	...

Figure 5.15: Knowledge representation using the same number of attributes for ports and OS.

All three options have pros and cons. The third proposal balances the features related to ports and OS, so clustering algorithms do not prioritize a group of features. The second proposal specifies a single port feature that sums up all the open ports on a system, so specific data about port scanning is not considered. In contrast, the number of attributes decreases in great extent, which is a good point. The first proposal includes all detected ports with their information. However the number of port features may overcome OS features if there are too many devices with different services on a network. In this case, results may be influenced only by port scan information. So it may be deduced that the third proposal is the most valid, as it equilibrates the number of features. But some drawbacks should be considered. First, information regarding the exact ports that have been detected is lost. Moreover, a bad selection of the port ranges would produce unsatisfiable clustering results. This selection can be arbitrarily assigned, or algorithms that calculate range limits proportionally to the number of open ports could be selected. In our experimentation, ranges have been arbitrarily selected but smaller ranges have been assigned to low values, due to the fact that the majority of open services are usually assigned to low ports. Finally, experiments have been conducted with these three different knowledge representations to validate them. They have been applied to the same data stored in *Consensus* after a security test.

Vulnerability data has been further analyzed to be included in the knowledge representation, as explained in Section 5.4.3. The final proposals are illustrated in figures 5.16, 5.17 and 5.18.

Open Ports					Possible OS						Security level			Risk Factor				
23	25	53	80	...	Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...	Note	Warning	Hole	None	Low	Medium	High	Critical
1	2	0	1	...	0.67	0.2	0	0	0	...	8	6	0	6	4	3	1	0

Figure 5.16: Knowledge representation using all detected ports, OS and vulnerability information.

Open Ports	Possible OS						Security level			Risk Factor				
Open ports	Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...	Note	Warning	Hole	None	Low	Medium	High	Critical
30	0.67	0.2	0	0	0	...	8	6	0	6	4	3	1	0

Figure 5.17: Knowledge representation using a single port range, OS and vulnerability information.

Port ranges				Possible OS						Security level			Risk Factor				
1 - 100	101 - 500	501 - 1024	...	Linux	Solaris	WinNT	WinXP SP1	WinXP SP2	...	Note	Warning	Hole	None	Low	Medium	High	Critical
25	4	0	...	0.67	0.2	0	0	0	...	8	6	0	6	4	3	1	0

Figure 5.18: Knowledge representation using port ranges, OS and vulnerability information.

5.4.5 Pattern implementation for clustering in Analia

The *AI Module* of *Analia* is in charge of creating the selected knowledge representation to customize the clustering input, as explained in Section 5.3.2. A View Manager has been implemented and included in *Analia*. This view manager helps creating different views that can be later used when executing clustering algorithms. As a result, knowledge representation management is independent from other clustering phases.

Security analysts can design OS views that specify which OS will be considered as features. Analysts can also design port range views where the inferior and superior values of every range can be set. Views with all the open ports or vulnerability types of selected devices can be generated as well. So the views explained in sections 5.4.1, 5.4.2 and 5.4.3 can be created and managed with the View Manager option of *Analia*. In addition, this View Manager generates the commands needed for the specific queries that will be applied to *Consensus* database. Moreover, these views can be combined before calling a clustering algorithm. Thus the views detailed in Section 5.4.4 can be also handled in the system.

An example of the View Manager is illustrated in Figure 5.19. This capture shows how to generate a simple view of port scanning, where all ports of the selected security scans will be included. Analysts only have to choose which security scans want to analyze and assign a name to that view. These scans are composed of the different tested devices. Therefore, the *AI module* will check all the open ports of the devices included in the selected security scans and will create the corresponding view. The queries to obtain all the information will be stored in the system and will be used when a clustering approach with that input view is selected to be executed.

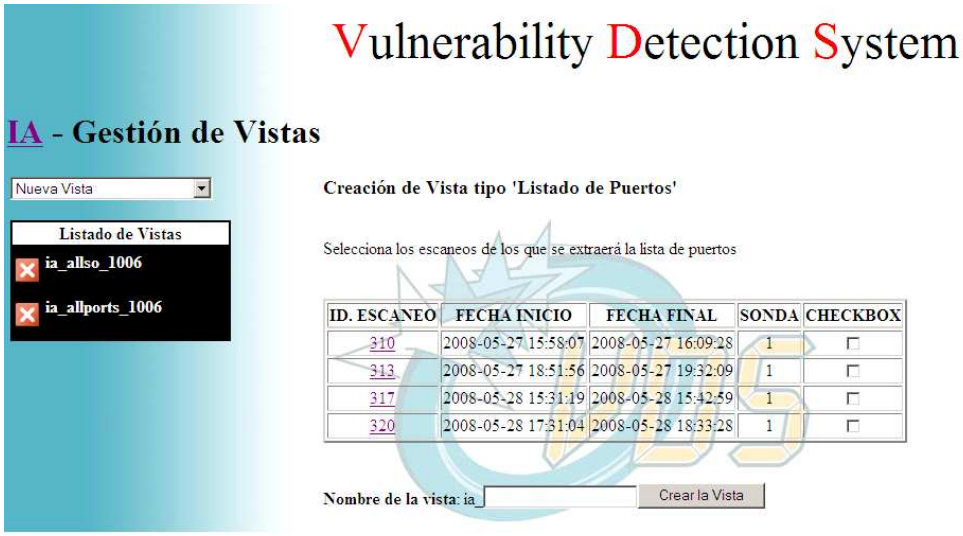


Figure 5.19: View manager to generate a pattern representation of open ports.

The View Manager implements the Algorithm 5.1. The output of this algorithm is the final query to be applied over the *Consensus* database. The *AI Control block* will run this algorithm when the input and control files have to be generated once a clustering approach has been selected (see Section 5.3.2).

Algorithm 5.1: *View Manager algorithm*

```
Let Let N = e_1, ..., e_n be a set of elements. An element is a tested device.
Let C = C_1, ..., C_k be a partition of N into subsets.
Let e_i = f_i1, ..., f_ip be an element, associated with a vector of p features.
Let S = S_1, ..., S_l be a set of l security scans. A scan is composed by different tested devices.
Let P = p_1, ..., p_m be a set of m ports.
Let R = r_1, ..., r_j be a set of j port ranges.
Let OS = os_1, ..., os_z be a set of z operating systems.
Let VI = sn, sw, sh, rfn, rfl, rfm, rfh, rfc be the set of security notes and risk factors.
Function View_manager(C: Training set, O: Option, name: View name) is
    switch O do
        case all ports
            S = Ask for security scans to analyze
            N = Select all devices from S
            forall e_i in N do
                P = Select their open or filtered ports
            Store the query (P, name)
            break
        case range of ports
            R = Ask for the port ranges to analyze
            Store the query (R, name)
            break
        case sum of all ports
            R = R_1, where R_1 = 1 - 65535
            Store the query (R, name)
            break
        case all operating systems
            S = Ask for security scans to analyze
            N = Select all devices from S
            forall e_i in N do
                OS = Select their operating systems
            Store the query (OS, name)
            break
        case vulnerability information
            Store the query (VI, name)
            break
    return query
```

The contribution of this View Manager allows separating two different processes: the creation of pattern representations and the execution of clustering algorithms. Simple and different knowledge representations can be created and stored very easily, independently of the clustering techniques. Later on, when configuring the parameters to call the clustering algorithm, these views can be combined to generate an input file with the features obtained from the merged views. So simple views are created once and then, at any time, analysts can choose which views to combine. Thus different input files can be created from the same database with the desired attributes, just changing the selected views.

5.5 Clustering in Analia

A clustering process usually includes the following steps, as explained in Section 2.3.2: pattern representation, pattern proximity definition, clustering, data abstraction and cluster validation. The previous section has detailed the pattern representation phase in *Analia*. This section describes the design and implementation of the clustering step, and also its integration in the *Analia* module. References to the definition of the pattern proximity measure will be included, as this phase is directly related to the clustering phase.

5.5.1 Clustering design in *Analia*

The main goal of applying an unsupervised learning technique like clustering in *Consensus* dataset is to find similarities and dissimilarities within information resulting from security tests. Many devices may have been tested and, therefore, many data may have been gathered. Then, the main challenge after collecting all this information is the extraction of conclusions from the dataset to facilitate the detection of irregular behaviors in networks or anomalous devices that could result in vulnerable systems. Thus clustering can help by grouping tested devices with similar vulnerabilities, so the same solution can be applied to a whole cluster to eliminate the existent vulnerabilities. Also unexpected changes in devices can be detected when they are surprisingly grouped in other clusters, and not with the rest of the devices that were supposed to be grouped. If these devices are grouped in other clusters, it means that several of their characteristics have been modified, by mistake or deliberately. So these devices can be more easily identified without having to check the characteristics of all the tested devices one by one, which could become an arduous task that would spend many resources and time. Even more, the time spent in checking all devices could be used by hackers to break into the system and exploit that vulnerability. Consequently, the less time spent in detecting a suspicious device, the less time needed to check it and correct its possible vulnerabilities and threats.

The analysis of the different clustering approaches detailed in Section 5.2.2 has concluded that the techniques most adequate for *Consensus* data set are K-means, X-means, SOM, Autoclass and CAOS. This is why these five approaches have been gradually included in *Analia* module. First studies were performed with K-means and X-means algorithms, due to their widely use in other environments and their adequacy and easiness to handle features obtained from tested devices. Results with these two techniques revealed good results when using port scanning and OS fingerprinting information of tested devices stored in *Consensus* dataset. Once the usefulness of clustering techniques was validated, this thesis focused on including other clustering approaches that could improve clustering results. This is why Autoclass and SOM were then deeply studied and included in *Analia*. CAOS was added to include multiobjetive benefits to *Analia*.

The *Clustering* module should be as modular and flexible as possible so as to ease the inclusion of new clustering algorithms whenever necessary. First experiments included only two clustering algorithms, whereas more algorithms have been integrated during this thesis completion. This is why several common criteria had to be previously defined in order to deal with the characteristics and configurations of the different clustering algorithms. The *Clustering* module should be also capable of handling output results of any of the clustering algorithms and then, format these results to be stored in *Consensus* database. Analysts should be able to look up clustering results using the same process, independently of the clustering algorithm called to obtain those results. Finally the output of these algorithms should be normalized to be better stored in *Consensus*.

The *Clustering* module is composed of three main blocks: *Input*, *Execution* and *Output* blocks. The state diagram of the *Clustering* module of *Analia* is shown in Figure 5.20. The different blocks and the state diagram of this module will be explained in the following paragraphs.

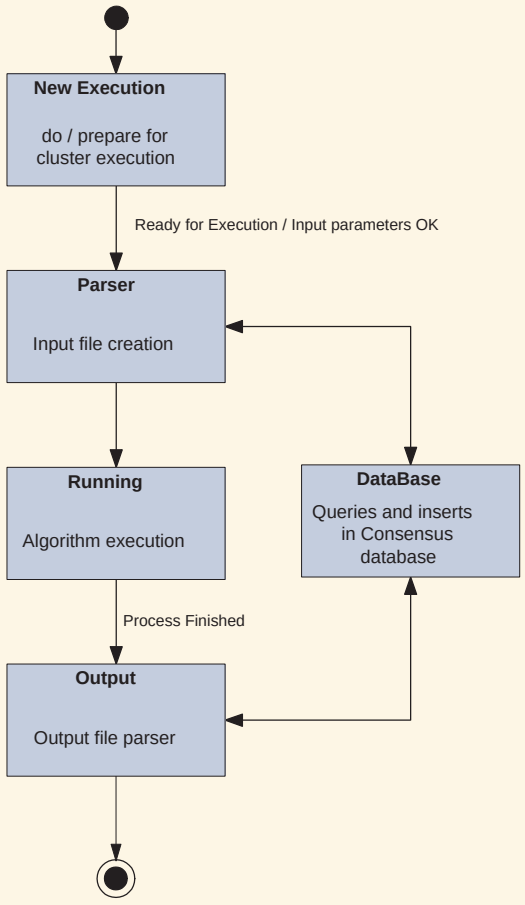


Figure 5.20: State diagram of the *Clustering* module of *Analia*.

5.5.1.1 Input block of *Analia* Clustering module

The *Input* block has to collect all the parameters needed to execute the clustering algorithm with the purpose of preparing the corresponding configuration file. Furthermore, this block has to elaborate the input file with the different samples to cluster, and the knowledge representation selected for the execution. The input parameters may vary depending on the clustering algorithm. Therefore, the input interface must consider the different possibilities. The parameters that must be configured before any cluster execution are the following:

- **Algorithm**: Different clustering approaches can be applied to the *Consensus* dataset. The different possibilities are: K-means, X-means, Autoclass, SOM and CAOS. Regarding this selection, other parameters will be needed to complete the configuration file.

- **View**: The view is related to the pattern representation of *Consensus* knowledge. As explained in Section 5.4, different knowledge representations can be created by using the *View Manager* tool of *Analia*. These simple knowledge representations can be combined to create a more complex pattern representation, including information not only from port scanning but also from OS fingerprinting or vulnerability search. The choice of the pattern representation must be performed in the *Input* block to create the input data file with the different samples represented by the selected features. So the view selection will determine the features to include in the input data file.

- **Security scans**: This parameter defines the input data to be clustered. Different security scans can be selected. A security scan contains several tested devices. Each of these devices will be considered an element or sample to be clustered. However, *Consensus* stores many data about each tested device and not all this data is suitable as input parameter for clustering algorithms. Then, the view will set which features finally are used to represent each sample. Data extracted from the *Security scans* parameter along with data obtained from the *View* parameter will be used to create the input data file, independently of the clustering algorithm that will process that file.

- **Number of clusters**: Some clustering algorithms demand for the definition of the number of clusters beforehand. This is the case of K-means and SOM. Autoclass can be configured to automatically predict the best number of clusters, or can be also given a predefined number of clusters to partition the dataset. Then, this parameter will be useful depending on the selected clustering approach.

- **Cluster range**: X-means requires specification of a range of values in which the K reasonably lies. So when selecting this clustering approach, the minimum value of K and the maximum value of K will have to be configured.

- **Seed**: This parameter fixes the starting value used by clustering algorithms to create random numbers to initialize their different processes. For example, K-means sets the centroid for each K cluster to random values at the beginning of the execution. The *Seed* parameter can influence in this initial process.

- **Maximum iterations**: When this parameter is configured, it can be used to limit the maximum number of iterations that a clustering algorithm will take to obtain a solution. When this parameter is not configured, clustering algorithms use their default values.

All these parameters are configured in the *New Execution* state of the *Clustering* module of *Analia* (see Figure 5.20). Security analysts can use the web interface of *Analia* to set them up. When these parameters are correctly configured for the desired clustering algorithm, the system transitions to the next state: the *Parser* state.

The *Parser* state creates the configuration file for the the selected clustering algorithm with the appropriate parameters. This configuration file may include the number of clusters or the cluster range to obtain, the configured seed, the maximum number of iterations, the name of the input data file or other parameters needed for the execution of the clustering approach. An example of a configuration file is shown in Figure 5.21, where the configuration file for a K-means execution with K=5 and a maximum of 60 iterations is set. This file also includes the name of the input data file and the name of the output file where the clustering algorithm will store the partition result.

```
@kcenters 5
@metric 2
@max_iteracions 60
@output_file /var/www/vds_analisis/AI/alg/xmeans/bin/temp/ia_198.out
@train_file /var/www/vds_analisis/AI/alg/xmeans/bin/temp/ia_198.arff
@test_file /var/www/vds_analisis/AI/alg/xmeans/bin/temp/ia_198.arff
```

Figure 5.21: Example of a configuration file of a K-means execution in *Analia*.

The *Parser* state has to build up the input data file where all the samples to be clustered are included. These samples stand for the tested devices. They are represented by the selected features chosen with the *View* parameter.

The input file is independent of the clustering algorithm, so new clustering approaches can be added to the system more easily at a later time. This is why the Attribute-Relation File Format (ARFF) [7] has been selected as the format for the input data file, due to the fact that it is a widely used file format to describe a list of instances sharing a set of attributes. ARFF files have two distinct sections: the Header information, and the Data information. The former contains the name of the relation, a list of the attributes and their types. The latter contains the value of the attributes for each of the samples and the expected class. In *Analia*, the Header information is built using the values configured in the *View* parameter, that define the attributes or features to be considered. On the other hand, the Data information is built using not only the *View* parameter but also the Security scans parameter. The combination of these values allows constructing the queries to *Consensus* database, where the tested devices to analyze are selected and the attributes to be interpreted are extracted.

Figure 5.22 shows an example of an ARFF file that includes 12 attributes or features. This file corresponds to the knowledge representation using a unique port range and all the operating systems (see Figure 5.14). This ARFF file shows only 4 instances or samples in the *@DATA* part, that correspond to 4 different tested devices. Each sample is in a different row and the values of the different features are separated by commas. A single class has been specified, due to this unsupervised learning environment where input information has not been classified before.

```
@RELATION vds
@ATTRIBUTE ports_totals NUMERIC
@ATTRIBUTE Solaris NUMERIC
@ATTRIBUTE FreeBSD NUMERIC
@ATTRIBUTE Linux_Kernel_2.4.X NUMERIC
@ATTRIBUTE Linux_Kernel_2.6.X NUMERIC
@ATTRIBUTE Microsoft_Windows_98 NUMERIC
@ATTRIBUTE Microsoft_Windows_NT NUMERIC
@ATTRIBUTE Microsoft_Windows_2000 NUMERIC
@ATTRIBUTE Microsoft_Windows_2000_Server NUMERIC
@ATTRIBUTE Microsoft_Windows_ME NUMERIC
@ATTRIBUTE Microsoft_Windows_XP NUMERIC
@ATTRIBUTE Microsoft_Windows_2003 NUMERIC
@ATTRIBUTE class {vds}

@DATA
8,0,0,0.7,0.7,0,0,0,0,0,0,0,vds
3,0,0,0,0,0,0,0.67,0.67,0,0.7,0.7,vds
3,0,0,0,0,0,0,0.67,0.67,0,0.7,0.7,vds
11,0,0,0.41,0.41,0,0,0,0,0,0,0,vds
...
```

Figure 5.22: Example of an ARFF input data file in *Analia*.

When the input file has been built, the system transitions to the Running state.

5.5.1.2 Execution block of *Analia* Clustering module

The *Execution* block corresponds to the *Running* state of the *Clustering* module. As its name specifies, this block is in charge of executing the configured clustering approach. Then, this block needs the ARFF input data file and the configuration file as input parameters. The *Algorithm* parameter will be used to select which clustering algorithm to run.

The clustering algorithms that can be executed have been detailed in Section 5.2.2. K-means and X-means have been implemented in the GRSI group. In fact, the original X-means algorithm was improved by the GRSI group to support missing values by refining the distance function and this improvement was named X-melan. Then, the included algorithm in *Analia* is the X-melan approach [28]. Regarding Autoclass, the included version is the Autoclass-C 3.3.4[1], which is a public domain version of AutoClass III, with some improvements from AutoClass X, implemented in the C language. Concerning SOM, the implementation used in *Analia* corresponds to the PhD of A. Fornells, also from the GRSI group [46]. The multiobjective approach is CAOS, a modification of the MOCK algorithm, included in the Diploma of Advanced Studies (DEA-*Diploma d'Estudis Avançats*) of A. Garcia [48], member of the GRSI as well. Other clustering approaches can be easily added to the system, when correctly formatting their inputs and outputs.

All the clustering algorithms have implemented the Minkowski metric with $p = 2$, that is the Euclidean distance, (see Equation 2.2), for the pattern proximity measure. This is one of the most widely used proximity functions. Nevertheless, other proximity measures can be included in the algorithms as well.

When the clustering execution has correctly finished, its output contains the input instances grouped in different clusters. Clustering results are stored in the output file, whose name has been previously set in the configuration file. This output file must be formatted to insert clustering results in *Consensus* database. The parsing of this output file and the storage of clustering results in the system database is made in the next state: the *Output* block.

5.5.1.3 Output block of *Analia* Clustering module

The *Output* block parsers the output clustering file in a regular format for all the clustering approaches. Then clustering results can be more easily added to *Consensus* database. This block is aligned with the *Output* state shown in Figure 5.20.

An example of a fragment extracted from an output file is shown in Figure 5.23. This file comprises the different obtained clusters. Each cluster has the number of elements that the clustering approach has grouped together, and lists the identifiers of each contained element. This identifier is related to its corresponding row in the input data file, that is to say, the order of the sample identifiers is the same as in the input file.

It can be seen from the input and output files that there is no relationship between the sample identifiers included in these files and the identifiers of the tested devices assigned in *Consensus* database. This is an advantage: clustering approaches can be used independently from *Analia* to cluster other datasets when the file formats are maintained. However, these results must be stored also in the *Consensus* framework, so they can be examined later and the rest of the phases of the clustering process can be applied to them. Hence, there must exist a connection between each sample identifier, that determines only the row position in the input file, and the identifier of that tested device in *Consensus*. This association was previously stored in the *Consensus* database during the *New execution* state of *Clustering* module. So it has to be retrieved in this *Output* block, in order to store the obtained partition with the corresponding device identifiers.

[1]Autoclass-C 3.3.4 - http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/autoclass-c-program.html

```
-----------------------------------------------------
            CLUSTER 0  (2 elements)
-----------------------------------------------------
        # Element: 13
        # Element: 17
-----------------------------------------------------
            CLUSTER 1  (3 elements)
-----------------------------------------------------
        # Element: 9
        # Element: 31
        # Element: 32
-----------------------------------------------------
            CLUSTER 2  (3 elements)
-----------------------------------------------------
        # Element: 24
        # Element: 27
        # Element: 29
-----------------------------------------------------
            CLUSTER 3  (6 elements)
-----------------------------------------------------
        # Element: 14
        # Element: 15
        # Element: 19
        # Element: 21
        # Element: 22
        # Element: 23
-----------------------------------------------------
.....
```

Figure 5.23: Fragment example of an output results file in *Analia*.

5.5.2 Clustering implementation in *Analia*

The first step in the *Clustering module* in *Analia* is the configuration of the execution parameters. This configuration is made via the web interface of *Analia*, where the different parameters are asked to analysts. An example of the configuration window is shown in Figure 5.24. The top of the window displays the different security scans to select, where the scan identifier, the init and end time of the scan, a description of the scan and the probe identifier that has gathered testing results are shown. The checkboxes allow selecting which are the scans, and consequently the tested devices to be converted in input samples for the clustering algorithm. The window center displays other important parameters, like the views to combine, the selected algorithm, the number of clusters to obtain, the maximum number of iterations and the seed to set up the initial clusters for the K-means algorithm, the one selected in Figure 5.24. A description can be introduced to insert a comment regarding that clustering execution.

When the execution of the selected algorithm is called from the web interface, a unique identifier is assigned to that execution and the configured parameters (description, algorithm, views, clusters, seed, maximum number of iterations and so on) are stored in *Consensus* database. This identifier is used to link the execution with the selected devices or elements that will be clustered. The

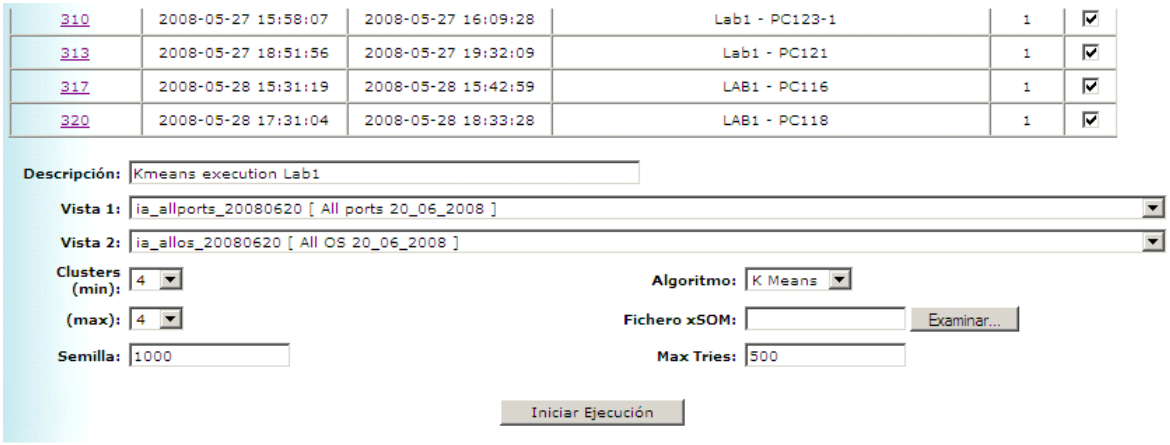


Figure 5.24: Configuration web interface of the clustering approaches in *Analia*.

input data and configuration files are created, and the corresponding clustering algorithm is finally executed.

When the clustering algorithm finishes, clustering results are shown to analysts. These results are obtained from the output file and the link stored in the database that connects the execution with the tested devices. Clustering results show each cluster and the devices grouped in the different clusters.

An example of a clustering results window is shown in Figure 5.25. This figure shows the IP address of the different tested devices that have been selected to analyze, and the cluster where they have been assigned. This example has grouped devices in five different clusters. At a glance, analysts can detect groups of devices with similar services and operating systems and, thus, devices with resembling vulnerabilities. Then primary efforts to solve a critical vulnerability can be concentrated in the devices grouped in the same cluster. The other clusters and, therefore, the rest of the devices can be analyzed later on when the most vulnerable devices have been already patched. Moreover, if analysts know in advance the characteristics of some tested devices, they can detect incongruences that can derive in finding computers with unexpected configurations because they had been misconfigured or manipulated. For example, if analysts expected to find a group of devices clustered together because they were supposed to share the same features, and finally, one of those devices is grouped in a different cluster, several conclusions can be extracted. This ungrouped device may have additional ports open, which means that it becomes susceptible to more vulnerabilities. Or maybe it has been misconfigured. Consequently, this error can be easily solved without having to analyze all the tested devices in detail.

Afterwards clustering results can be consulted not only at the end of the execution but at any time, due to the fact that they are stored in *Consensus* database. The list of executions is shown to analysts, who can select which one to visualize to check previous results. An example of a list of executions is shown in Figure 5.26, where the clustering approach used to obtain results, its description, the configured views, seed and number of clusters for each execution, are listed. When consulting detailed clustering results, other system features like cluster validation or explanation of clustering results can also be checked. The characteristics of each of the tested devices can be examined too, as this information is still stored in the central database.

Vulnerability Detection System

Lista de Clusters

Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
172.16.1.158	84.88.232.227	172.16.1.159	172.16.1.25	10.0.14.130
172.16.1.30	84.88.232.5	10.0.14.203	172.16.1.28	10.0.14.131
172.16.1.201	84.88.232.238	84.88.232.223	172.16.1.203	10.0.14.132
172.16.1.160	84.88.232.123	10.0.14.202		10.0.14.133
10.0.14.237	84.88.232.180	10.0.14.125		10.0.14.134
	84.88.232.85	10.0.14.124		10.0.14.135
	172.16.1.89	10.0.14.206		10.0.14.136
	172.16.1.17	10.0.14.202		10.0.14.137
	172.16.1.161			10.0.14.138
	172.16.1.93			10.0.14.139
	172.16.1.29			10.0.14.140
	172.16.1.88			10.0.14.141
	172.16.1.121			10.0.14.142
	172.16.1.140			10.0.14.143
	172.16.1.154			10.0.14.150
	84.88.232.160			10.0.14.151
	84.88.232.190			10.0.14.152
	172.16.1.164			10.0.14.153
	10.0.5.11			10.0.14.154
	10.0.5.12			10.0.14.155
	10.0.5.13			10.0.14.156
	10.0.5.14			10.0.14.157
	10.0.5.15			10.0.14.158
	10.0.5.14			10.0.14.157
	10.0.5.15			10.0.14.158
	10.0.5.16			10.0.14.159
	10.0.14.203			10.0.14.160
	10.0.14.207			10.0.14.161
	10.0.14.208			10.0.14.150
				10.0.14.151
				10.0.14.152
				10.0.14.153
				10.0.14.154
				10.0.14.155
				10.0.14.156
				10.0.14.157
				10.0.14.158
				10.0.14.159
				10.0.14.160
				10.0.14.161
				172.16.1.115
				10.0.14.205
				10.0.14.204
				10.0.14.107
				10.0.14.123
				10.0.14.121
				10.0.14.116
				10.0.14.118

Figure 5.25: Example of a clustering result in *Analia*.

Algoritmo	Vista	Semilla	Clusters
kmeans	ia_allports_20080620, ia_allos_20080620 Test 1 Ports-OS Seed=1 K=5	1	[5,5]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 5 OS Seed=10000 K=6	10000	[6,6]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 4 OS Seed=1000 K=6	1000	[6,6]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 3 OS Seed=100 K=6	100	[6,6]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 2 OS Seed=10 K=6	1	[6,6]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 1 OS Seed=1 K=6	1	[6,6]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 5 OS Seed=1	1	[5,5]
kmeans	ia_allos_20080620, ia_allos_20080620 Test 4 OS Seed=10000	10000	[5,5]

Figure 5.26: Example of the web interface with a list of clustering executions in *Analia*.

5.6 Cluster validation in Analia

Nowadays there are a number of validity indices available to evaluate clustering results, but none of the analyzed methods in Section 5.2.3 use different features for classifying and assessing. This section presents a contribution in the cluster validation phase of *Analia* to computer-aided assess the clustering results from *Consensus* datasets. New validation indices are proposed in this thesis: the *Cohesion factors* [26]. This new method is a cluster validity index to establish a relationship between the results obtained after applying a clustering technique and their utility in the vulnerability detection analysis. *Cohesion factors* use not only the features included in the clustering phase, but also other information already stored in *Consensus* database after performing a security assessment.

Cluster validity indices can be used to compare the performance of different clustering results for different number of clusters or parameters. To select the optimal quantity of clusters, the clustering process can be applied once for each number of clusters, and then validity indices results are compared. In addition, the best results obtained from different clustering techniques can be selected by comparing their associated validity indices. So these indices provide insights into the problem of predicting the correct number of clusters, which was one of the drawbacks of some of the proposed clustering methods when applied to a completely unsupervised domain. Therefore, the *Cohesion factors* are proposed to improve the prediction about the number of relevant clusters in the *Consensus* domain.

Each sample of the input clustering dataset corresponds to a tested device. Firstly, the features that represent each sample are the detected open ports and possible operating systems. All of them are numerical features that comply with the format of input attributes that clustering algorithms accept. Moreover, this data is critical for security analysts. The open ports identify the available services in a device and, consequently, the possible vulnerable targets of that device. The operating system also affects to determine possible vulnerabilities, due to the fact that different operating systems suffer from different flaws. Besides, information directly related to vulnerability testing can be acquired by different testing tools. However these tools provide this information in their own format. This information tends to be represented in long sentences that define the vulnerability type and the affected service. Depending on the vulnerability, its risk factor and some identifier like the CVE (Common Vulnerabilities and Exposures) [87] may be included, but these values are not always incorporated. Thus, it is troublesome to parametrize this information to be formatted as input features of the clustering algorithms.

Clustering results in *Analia* show partitions where each cluster contains tested devices with similar open ports and operating systems. The main goal of this contribution is to demonstrate that the samples that have been grouped in the same cluster also share similar vulnerabilities. Furthermore, this contribution will help to find a proper value of the number of clusters in this domain. This is why two different *Cohesion factors* are proposed. These indices are based on the validity indices explained in Section 5.2.3. These two *Cohesion factors* are named **Intracohesion factor** and **Intercohesion factor**.

5.6.1 Intracohesion factor

The *Intracohesion factor* evaluates the cohesion between the elements of a cluster, that is to say, it measures the compactness of each cluster. This factor is a distance measure on the set of data points in a cluster C_k. First, the *Similarity (S)* between two elements of a cluster is calculated by adding the vulnerabilities common to both elements. This value is divided by the number of vulnerabilities of one of the compared elements, as shown in Equation 5.1. Subsequently, the *Intracohesion factor* of a cluster is calculated by adding all the similitudes of that cluster, calculated for all the pairs of elements of that cluster, and then divided by the number of different pairs of elements of the

same cluster. This formula is shown in Equation 5.2. Finally, the *Mean Intracohesion factor* is the average of the *Intracohesion* obtained from every cluster in the system. It is desirable for the clusters to be as compact as possible, as it means that elements of a cluster have a high degree of similarity inside that cluster. Consequently, clusters that maximize the *Intracohesion factor* and the *Mean Intracohesion factor* are preferred.

$$S(x_i, x_j) = \frac{Common \quad Vulnerabilities(x_i, x_j)}{Total \quad Vulnerabilities(x_i)} \tag{5.1}$$

$$Intracohesion = \frac{\sum_{x_i \in C_k} \sum_{\substack{x_j \in C_k \\ x_j \neq x_i}} S(x_i, x_j)}{|C_k|^2 - |C_k|} \tag{5.2}$$

$$Mean\ Intracohesion = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{x_i \in C_k} \sum_{\substack{x_j \in C_k \\ x_j \neq x_i}} S(x_i, x_j)}{|C_k|^2 - |C_k|} \tag{5.3}$$

Table 5.1 shows a calculation example of similarities between the different elements of a cluster. Security analysts can refer to this information, and check the *Intracohesion factor* for each cluster and the global *Mean Intracohesion factor*. These values reveal how similar are the devices of a cluster. They are also useful to detect devices that, in normal conditions should be grouped in a certain cluster, and due to improper changes in their services, they appear grouped with other elements.

Cluster n	**Device 1**	**Device 2**	**Device 3**	**Device 4**
Device 1	–	S(1,2)	S(1,3)	S(1,4)
Device 2	S(2,1)	–	S(2,3)	S(2,4)
Device 3	S(3,1)	S(3,2)	–	S(3,4)
Device 4	S(4,1)	S(4,2)	S(4,3)	–

Table 5.1: Example of a cluster matrix that includes Similarity values between tested devices.

5.6.2 Intercohesion factor

The *Intercohesion factor* evaluates the separation between clusters. In fact, it evaluates the similarity between clusters, so low values of this factor are preferred. Different clusters should be widely spaced and as less similar as possible, so that elements of different clusters do not share vulnerabilities. Then, the *Distance (D)* function between two different clusters must be calculated. Firstly, the coincidence of vulnerabilities that all the elements of one cluster have in common is computed. Afterward, D checks the coincidence of common vulnerabilities between two clusters and divides this value by the number of vulnerabilities that all the elements of one of the clusters have in common. The function D is defined as shown in Equation 5.4. Finally, the *Distance* is calculated for all the obtained clusters, so that the *Intercohesion* is the average of distances calculated for every pair of different clusters. Equation 5.5 shows the *Intercohesion factor* formula.

$$D(C_i, C_j) = \frac{Common \quad Vulnerabilities\ (C_i, C_j)}{Common \quad Vulnerabilities\ (C_i)} \tag{5.4}$$

$$Intercohesion = \frac{\sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{K} D(C_i, C_j)}{K^2 - K} \tag{5.5}$$

Table 5.2 shows a calculation example of the distances between the different clusters obtained in a execution. Security analysts can look up this information, including the *Intercohesion factor*.

These values show analysts how different are the resultant clusters between them. Consequently, if two clusters are widely spaced they should group elements with very different vulnerabilities.

	Cluster 1	**Cluster 2**	**Cluster 3**
Cluster 1	–	D(1,2)	D(1,3)
Cluster 2	D(2,1)	–	D(2,3)
Cluster 3	D(3,1)	D(3,2)	–

Table 5.2: Example of a comparison matrix of distances between clusters of a partition.

Cohesion factors are a useful tool to guide unsupervised analysis of data obtained from a security assessment. Best results of clustering should include high *Intracohesion* and low *Intercohesion* values. Moreover, the utility of these factors can be applied to improve the estimation of the number of clusters represented in the data. This improvement will be explained in Section 5.6.3. Therefore, *Cohesion factors* can be used to calculate the optimal number of K for the K-means algorithm [26]. They can also be used to select the execution with the most preferable number of clusters in the SOM approach. These two methods can not automatically obtain the best number of partitions, so this contribution is particularly relevant for the estimation of robust clustering partitions in techniques that require the definition of the number of clusters beforehand. In addition, these factors can be useful to compare different clustering solutions with the optimal number of clusters in order to obtain the partition with the best index rate.

5.6.3 Prediction of the number of clusters in Analia

The prediction of the correct number of clusters is a fundamental problem in unsupervised classification problems. Many clustering algorithms require the definition of the number of clusters beforehand like, for example, K-means [60] or SOM [72] (see Section 2.3.2). To overcome this problem, various cluster validity indices have been proposed to assess the quality of a clustering partition, like *C-Index* [63], the (DB) index [33], the *Dunn* index [40] or *Silhouette* [103] index (see Section 2.3.4). This approach consists of running a clustering algorithm several times and obtaining different partitions. Then, the clustering partition that optimizes a validity index is selected as the best partition.

Cohesion factors can be also used as a measure to select the best partition and hence predict the number of clusters appropriate for the analyzed data set. When having executed a clustering algorithm several times with different parameters, or different clustering algorithms, several clustering solutions are obtained. Then *Cohesion factors* are calculated for each clustering solution. The best solution should score the highest *Intracohesion factor* and the lowest *Intercohesion factor*. If *Intracohesion* is high, every element is very similar to the elements of the same cluster. If *Intercohesion* is low, clusters are separated. When having more clusters for a data set, cluster size decreases as there are more groups to insert the elements; therefore, it is easier that clusters become more compacted. However, the more number of clusters, the more possibilities that two clusters become more similar because elements' differences are not so significant to create so many clusters. Thus a compromise between both cohesion metrics should be achieved and this compromise will fix the best number of clusters for this data set. Besides, the difference between these two values should be as high as possible. This solution will provide the best partition with the optimal number of clusters for these indexes.

Other validity indices have been integrated in *Analia* too. So they can be used to select the best execution over a set of executions. In this case, high values for *Dunn* and *Silhouette* indexes are preferred, whereas low values for *DB* index are preferred. Analysts can choose any of the indexes to select the best execution from a set of clustering results, and obtain the partition with the best number of clusters.

The process to get the best partition, and hence the most correct number of clusters in *Analia* follows the Algorithm 5.2. The global process is summarized in the following steps:

1. Select the executions to analyze.
2. Calculate validation indices for each execution.
3. Select the validation index as decision criteria.
4. Obtain the best execution with the best number of clusters.

Algorithm 5.2: *Algorithm for the selection of the best number of clusters in Analia*

```
Let N = e_1, ..., e_n be a set of elements. An element is a tested device.
Let C = C_1, ..., C_k be a partition of N into subsets.
Let E = E_1, ..., E_i be a set of executions, where each execution corresponds to a different partition C.
Let Intra = Intra_E_1, ..., Intra_E_i be a set of Intracohesion factors.
Let Inter = Inter_E_1, ..., Inter_E_i be a set of Intercohesion factors.
Let Dif_Cohesion = Dif_Cohesion_E_1, ..., Dif_Cohesion_E_i be a set of difference between Cohesion factors.
Let Dunn = Dunn_E_1, ..., Dunn_E_i be a set of Dunn factors.
Let DB = DB_E_1, ..., DB_E_i be a set of DB factors.
Let Silhouette = Silhouette_E_1, ..., Silhouette_E_i be a set of Silhouette factors.
Function Best_Execution(E: Executions set, O: Option) is
    switch O do
        case Cohesion factors
            forall E_i in E do
                Intra_E_i = Calculate_Intracohesion_Factor(E_i)
                Inter_E_i = Calculate_Intercohesion_Factor(E_i)
                Dif_Cohesion_E_i = Intra_E_i − Inter_E_i
            best_C = E_i ∈ E | Intra_E_i is maximum and Inter_E_i is minimum and Dif_Cohesion_E_i is
            maximum
            break
        case Dunn
            forall E_i in E do
                Dunn_E_i = Calculate_Dunn(E_i)
            best_C = E_i ∈ E | Dunn_E_i is maximum
            break
        case DB
            forall E_i in E do
                DB_E_i = Calculate_DB(E_i)
            best_C = E_i ∈ E | DB_E_i is minimum
            break
        case Silhouette
            forall E_i in E do
                Silhouette_E_i = Calculate_Silhouette(E_i)
            best_C = E_i ∈ E | Silhouette_E_i is maximum
            break
        case Combination
            forall E_i in E do
                Intra_E_i = Calculate_Intracohesion_Factor(E_i)
                Inter_E_i = Calculate_Intercohesion_Factor(E_i)
                Dif_Cohesion_E_i = Intra_E_i − Inter_E_i
                Dunn_E_i = Calculate_Dunn(E_i)
                DB_E_i = Calculate_DB(E_i)
                Silhouette_E_i = Calculate_Silhouette(E_i)
            best_C = WeightedVotingScheme(E, Intra, Inter, Dif_Cohesion, Dunn, DB, Silhouette)
            break
    return best_C
```

Single indices can be selected as decision criteria, not only the most applied validity indices (*Dunn*, *DB* or *Silhouette* indices) but also the *Cohesion factors* designed *ad hoc* for this security domain. Due to the fact that security analysts do not usually care about the selected index criteria, but for the best clustering solution, an automated mechanism has been designed to combine all the calculated validity factors. This mechanism is based on a weighted voting scheme. This scheme ranks the list of options based on the number of votes each option earns, considering that some votes carry more weight than others. In this case, *Cohesion factors* are assigned a higher weight than the rest of the other indices, in order to obtain the best solution based on all indices but being more influenced by *Cohesion factors* results. Then, security analysts can easily get the best partition from a list of executions without any previous knowledge about validity indices, knowledge not usually related with their study area.

5.6.4 Implementation of cluster validation in Analia

The cluster validation process has been implemented in a compact module, that complies with the following requisites: its design must facilitate the integration with the other parts of the *Analia* module in so as to dispose of all the clustering cycle in the system; besides, it should be as much autonomous as possible, so it can be used independently in other environments to calculate validity indexes of other data sets; its design should permit the inclusion of new validity indexes without trouble; and, obviously, its results should be correctly presented. Finally, a new module called *Indexus* has been designed and integrated in the *AI Control block* of *Analia*.

Indexus system is composed of three main blocks: *Indexus*, *Parser* and *Index Calculation*. The *Indexus* block is the core module. It is in charge of coordination the input data set, the indices computation, and the storage of the calculation results in the central database. The *Parser* block compiles the input samples and the clustering results in order to obtain the partitioned data set. Finally, the *Index Calculation* module has to compute all the calculations necessary to obtain the different validity indices. The block diagram of the *Indexus* module is shown in Figure 5.27.

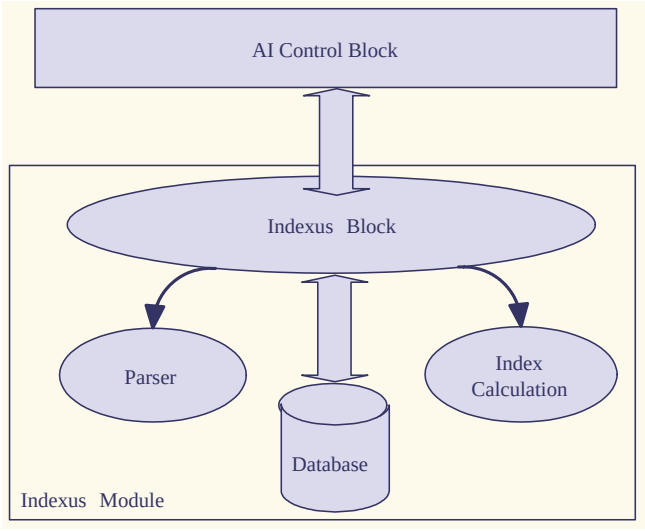


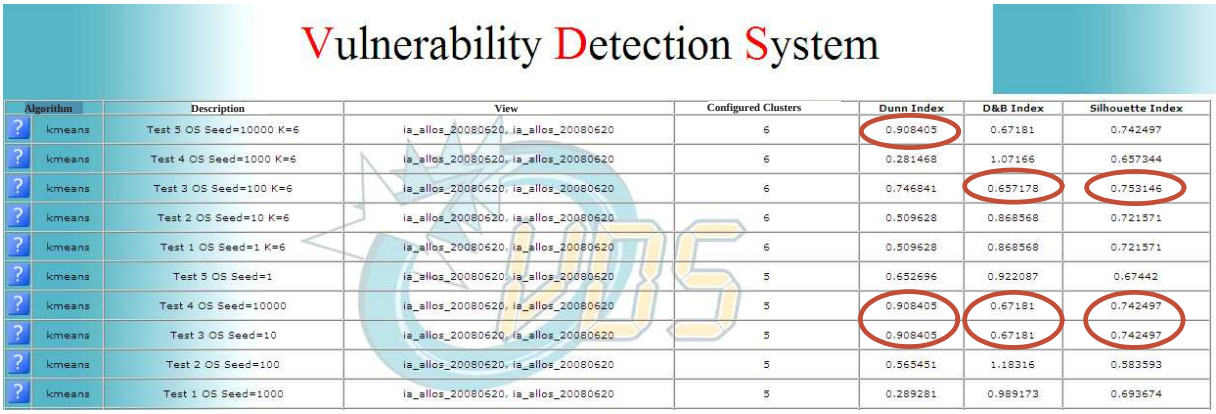
Figure 5.27: Block diagram of the Indexus module.

Indexus system must be applied over a data set when the clustering algorithm has partitioned its samples into different clusters. Then the inputs of *Indexus* are three different files: the configuration file, the data file and the clustering file. The configuration file contains the number of clusters, features and samples, the name of the output file to store validity index results, the name of

the data file and the name of the clustering file. The data file complies with the ARFF format [7], and it is the same file previously created as input for the clustering algorithm. Finally, the clustering file contains the resultant partition of the different samples in their respective clusters. This clustering file is created from the results stored in *Consensus* after any of the clustering approaches has been applied on a data set of tested devices. The validity process is completely independent of the clustering process by using this file. This validity process is applied once the data set has been partitioned, independently of the selected clustering approach. Results from any clustering approach are stored using the same format and, therefore, the use of this information is transparent for the subsequent phases. Moreover, if these three files are created from other data sets in other environments preserving the same file formats, *Indexus* can be independently called using its command line interface to calculate their validity indices.

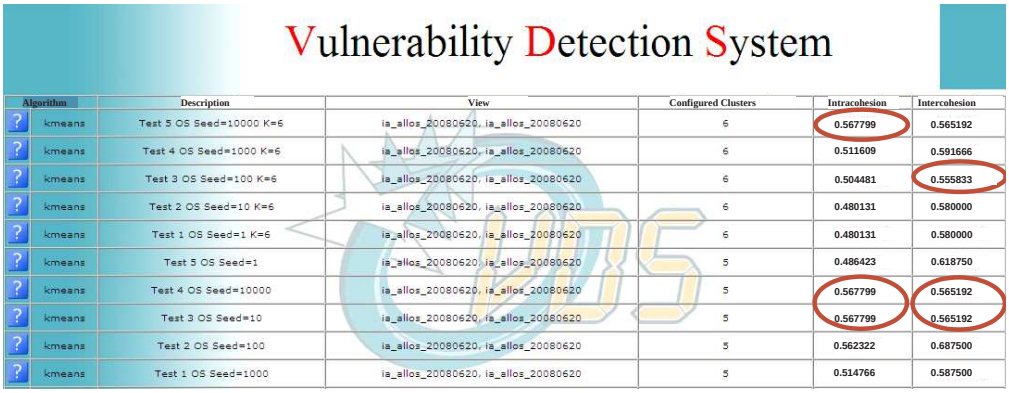
The *Parser* block contains all the functions related to the reading of the configuration, data and clustering files. Finally, the *Index Calculation* block includes the different functions to compute the different validity indices: *Dunn*, *DB*, *Silhouette* and *Cohesion* indices.

When the different indices have been calculated, they are stored in the output file, and also in the central database of *Consensus*, so they can be viewed with the *Analia* web interface. An example is shown in Figure 5.28. This screenshot pictures a table where each row corresponds to a different clustering execution. The columns show the characteristics of each execution: the applied clustering approach, a description of the execution, the used dataset, the number of clusters configured for each execution, and the value of the different validity indices for each execution. It can be seen that all executions have applied the K-means algorithm over the same dataset, but the number of configured clusters and seeds are variable. The best index values have been circled in red. Security analysts can perform their own analysis by selecting the execution with the best scoring. Regarding *Dunn* index, three executions score the same maximum value. Two of them are configured for 5 clusters, whereas the other one is configured for 6 clusters. However, when consulting clustering results (the question mark of the first column allows visualizing the obtained partition with the different devices distributed in their corresponding clusters), all three partitions are identical and the one configured with 6 clusters contains an empty cluster. Regarding *DB* and *Silhouette* indices, they coincide with *Dunn* index by selecting the same best executions for 5 clusters. On the other hand, a different execution for 6 clusters is considered the best one for these two indices. Figure 5.29 shows a screenshot with the same executions of the previous figure, but considering *Cohesion factor* values. The same conclusions can be extracted from this figure, regarding the executions with 5 and 6 clusters. High *Intracohesion* and low *Intercohesion* values are preferred. Then, analysts would have different executions to select.



Algorithm	Description	View	Configured Clusters	Dunn Index	D&B Index	Silhouette Index
kmeans	Test 5 OS Seed=10000 K=6	ia_allos_20080620, ia_allos_20080620	6	0.908405	0.67181	0.742497
kmeans	Test 4 OS Seed=1000 K=6	ia_allos_20080620, ia_allos_20080620	6	0.281468	1.07166	0.657344
kmeans	Test 3 OS Seed=100 K=6	ia_allos_20080620, ia_allos_20080620	6	0.746841	0.657178	0.753146
kmeans	Test 2 OS Seed=10 K=6	ia_allos_20080620, ia_allos_20080620	6	0.509628	0.868568	0.721571
kmeans	Test 1 OS Seed=1 K=6	ia_allos_20080620, ia_allos_20080620	6	0.509628	0.868568	0.721571
kmeans	Test 5 OS Seed=1	ia_allos_20080620, ia_allos_20080620	5	0.652696	0.922087	0.67442
kmeans	Test 4 OS Seed=10000	ia_allos_20080620, ia_allos_20080620	5	0.908405	0.67181	0.742497
kmeans	Test 3 OS Seed=10	ia_allos_20080620, ia_allos_20080620	5	0.908405	0.67181	0.742497
kmeans	Test 2 OS Seed=100	ia_allos_20080620, ia_allos_20080620	5	0.565451	1.18316	0.583593
kmeans	Test 1 OS Seed=1000	ia_allos_20080620, ia_allos_20080620	5	0.289281	0.989173	0.693674

Figure 5.28: Example of the cluster validity process in *Analia*.



Vulnerability Detection System

Algorithm	Description	View	Configured Clusters	Intracohesion	Intercohesion
kmeans	Test 5 OS Seed=10000 K=6	ia_allos_20080620, ia_allos_20080620	6	0.567799	0.565192
kmeans	Test 4 OS Seed=1000 K=6	ia_allos_20080620, ia_allos_20080620	6	0.511609	0.591666
kmeans	Test 3 OS Seed=100 K=6	ia_allos_20080620, ia_allos_20080620	6	0.504481	0.555833
kmeans	Test 2 OS Seed=10 K=6	ia_allos_20080620, ia_allos_20080620	6	0.480131	0.580000
kmeans	Test 1 OS Seed=1 K=6	ia_allos_20080620, ia_allos_20080620	6	0.480131	0.580000
kmeans	Test 5 OS Seed=1	ia_allos_20080620, ia_allos_20080620	5	0.486423	0.618750
kmeans	Test 4 OS Seed=10000	ia_allos_20080620, ia_allos_20080620	5	0.567799	0.565192
kmeans	Test 3 OS Seed=10	ia_allos_20080620, ia_allos_20080620	5	0.567799	0.565192
kmeans	Test 2 OS Seed=100	ia_allos_20080620, ia_allos_20080620	5	0.562322	0.687500
kmeans	Test 1 OS Seed=1000	ia_allos_20080620, ia_allos_20080620	5	0.514766	0.587500

Figure 5.29: Example of the cluster validity process with Cohesion factors in *Analia*.

From AI and data analysis point of view, Chapter 6 details that more exhaustive analysis of index results have been performed, and Student's t-tests have been applied over these results to know whether differences between index results are statistically significant or not. On the other hand, from security standpoint, this analysis is not so relevant due to the fact that different partitions of the same dataset can give different points of view. These options can be useful to analysts to detect hidden patterns in the tested devices, and can enrich analysis results. Moreover, if analysts do not want to perform this previous analysis, an automatic function permits selecting the best executions from a group of executions without having to manually study the different validity index results.

When selecting an execution, analysts can see the clustering results where each tested device is included in a cluster. An example of a cluster matrix that includes the similarity between devices of a cluster, as explained in Table 5.1, is shown in Figure 5.30. This figure shows a cluster with 14 devices from a tested lab (from 10.0.14.130 to 10.0.14.143), where 19 vulnerabilities have been detected in most of the devices, and 20 vulnerabilities have been detected in the two first devices of this cluster. Each cell contains the Similarity value between the devices of the corresponding row and column. In most of the cells, the Similarity value equals to 1 (19/19). This means that those devices have the same 19 vulnerabilities in common. The first two columns have similarities equal to 19/20, which means that 19 of the 20 detected vulnerabilities are the same in both devices. The high value of the Intracohesion factor of this cluster shows that it is a cohesive cluster, which means that the devices of this cluster share almost the same vulnerabilities.

Cluster 4

	10.0.14.130	10.0.14.131	10.0.14.132	10.0.14.133	10.0.14.134	10.0.14.135	10.0.14.136	10.0.14.137	10.0.14.138	10.0.14.139	10.0.14.140	10.0.14.141	10.0.14.142	10.0.14.143
10.0.14.130	20/20	20/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.131	20/20	20/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.132	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.133	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.134	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.135	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.136	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.137	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.138	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.139	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.140	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.141	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.142	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19
10.0.14.143	19/20	19/20	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19

Cluster Intracohesion = 0.886667047114

Figure 5.30: Example of a cluster matrix with similarity values between tested devices and its Intracohesion cluster value in *Analia*.

Analysts can check the comparison matrix of distances between clusters. An example is shown in Figure 5.31, where the data set has been divided in 5 clusters. This matrix shows how different are the obtained clusters between them as explained in Table 5.2. For example, the first column is related to *Cluster 0*. The denominator of the fractions shown in this column equals to 1, which means that all the elements of the *Cluster 0* only have 1 vulnerability in common. It can be seen that the numerator of the fractions of the same column equals to 1 too, which means that all the other clusters also have the same vulnerability in their devices. So it can be deduced that this is a very common vulnerability. Regarding the last column, it is referred to *Cluster 4*. Its elements share 13 vulnerabilities and, from all of them, only 1 vulnerability is shared with *Cluster 0*, as the numerator of the last cell of the first row shows. In fact, the values shown in this matrix should be as low as possible, as each cell represents how many vulnerabilities have the corresponding clusters in common. Besides, low values of the Intercohesion factor are preferred.

From the Intercohesion cluster matrix, analysts can deduce which are the clusters with more vulnerabilities. In the example shown in Figure 5.31, the cluster that contains more vulnerabilities in common between its devices is *Cluster 4*, with 13 vulnerabilities (check the denominator shown in its column). In contrast, the cluster with less vulnerabilities in common is *Cluster 0*. However it is very important to check if these vulnerabilities are critical or not. It is possible that a cluster with only one critical vulnerability becomes more dangerous than a cluster with several low vulnerabilities. So this risk factor will have to be considered when deciding which problems or vulnerabilities must be firstly solved. The good point is that devices with that critical vulnerability will be probably grouped in the same cluster, easing the task to security analysts.

Cluster\Cluster	0	1	2	3	4
0	--	1/4	1/10	1/4	1/13
1	1/1	--	2/10	2/4	3/13
2	1/1	2/4	--	4/4	7/13
3	1/1	2/4	4/10	--	4/13
4	1/1	3/4	7/10	4/4	--

Intercohesion factor = 0.565192307692

Figure 5.31: Example of a matrix with distance values between clusters of a partition and its Intercohesion value in *Analia*.

Finally, security analysts can choose different mechanisms to select the best clustering executions. A manual selection can be performed when analysts want to study different clustering results. Alternatively, the system can select the best partitions from a group of executions and show them to analysts. All of them are based on validity indices. They can be used to detect not only which partitions obtain better scoring regarding validity results but also which is the best number of partitions to divide a given data set. All of these mechanisms are integrated in *Analia*, so an enriched clustering process is included in this framework providing a new contribution of this thesis [28, 47].

5.7 Explanation of results in Analia

Classifications obtained by data clustering methods are often difficult to interpret conceptually. In most cases, they require an additional interpretation since no conceptual description is provided. This point has been already corroborated in *Analia*, due to the fact that results provided by algorithms like K-means, X-means, Autoclass, MOCK and SOM group devices according to data attributes but do not explain the reason of that grouping. Afterwards a further interpretation is needed to give an accurate explanation of the clustering results to network security experts. This section presents a new approach to explain clustering results adapting the Anti-unification algorithm. Two different contributions are detailed and integrated in *Analia*. The first contribution adapts the Anti-unification algorithm to the peculiarities of the security testing dataset of *Consensus*, providing descriptions that explain clustering results to security analysts [23]. The second contribution improves the first adaptation by supplying more detailed descriptions of every obtained cluster [24]. This improved version is called DAU. Both contributions are explained in next sections.

5.7.1 The Anti-unification in Analia

When clustering in a network security environment, not only the distribution of tested devices in groups regarding their vulnerabilities is useful information for security analysts but also the reason of that organization. This thesis proposes a value-added service by including symbolic descriptions that explain why a subset of cases has been grouped together. These descriptions are based on the Anti-unification concept introduced in [8], although with some differences.

The Anti-unification of two objects is defined as the most specific generalization of both objects. It is a description with the attributes shared by both objects whose value is the most specific one. The main goal of Anti-unification is to obtain a new element that contains all the common features of the different elements of a cluster. Any feature not shared between all the elements of a cluster will not be added to this new element. This new element will represent the whole cluster to explain why these elements have been grouped together.

Let C be a cluster and let $e_1,\ldots,e_n$ be the set of elements of that cluster after the application of a clustering algorithm included in *Analia*. It does not matter whether clustering results have been calculated using K-means, X-means, SOM, Autoclass, CAOS or any other new approach, as results from all of them have been processed to be stored in *Consensus* using the same format.

Each element e_j is described by a set of attributes A, where each attribute $a_k \in A$ takes values in V_k. The Anti-unification algorithm (AU) builds a new feature term D to describe the cluster C. The proposal is based on describing a cluster C using a symbolic description D, with the following rules:

- D contains the common attributes to all the elements in cluster C. Those attributes with unknown value in some element $e_j \in C$ are not considered in constructing D.
- Let a_k be an attribute common to all the elements in C taking values on a set V_k. The attribute a_k is not included in the description of D when the union of the values that a_k takes in the elements in C is exactly V_k.

Finally, the Anti-unification of the elements of a cluster C is a description D that contains the common attributes to all the elements in C. Those attributes with unknown value in any element $e_j \in C$ or those attributes that do not appear in some element $e_j \in C$ are not considered in constructing D. An attribute will not be either included in D when the union of the values that the attribute takes in the elements is the set of all possible values. The main steps of AU adapted to *Analia* domain are shown in Algorithm 5.3.

Algorithm 5.3: AU algorithm adapted to *Analia* domain

```
Let E be the set of elements of a cluster C
Let A be the set of attributes describing the elements of E
Let V_k be the set of all the possible values of the attribute a_k
Let D be the antiunification of the elements of E
Function AU is
    forall a_k in A do
        if a_k = v for all e_i in E then
            a_k takes value v in D
        Let v_ki be the value that each e_i ∈ E holds in a_k
        if union(v_ki) = V_k then
            a_k will not appear in D
        if v_ki = null value then
            a_k will not appear in D
        Otherwise a_k = union(v_ki in D)
    return D
```

An example of AU adaptation over *Analia* dataset is shown in Table 5.3. Let C be a cluster of four elements, where e_1, e_2, e_3 and e_4 are network devices whose attributes detail their open ports, possible operating systems and the number of detected security notes. Let D be a symbolic description of the cluster C. Attributes related to ports 25, 53 and 80 are not included in description D due to the fact that some elements do not have any value assigned to them. Table 5.3 shows that attributes related to ports 21, 135 and 445 are present on all elements and their value is always '1'. This value means that these ports are all open in all the elements of the cluster. As these attributes have the same value, this value is also reflected in D. Regarding OS, only the attribute 'W2003 Server' has the same value in all elements and hence this feature will have its value included in D. The rest of the attributes appear in all elements but with different values; thus those attributes will be included in D with the union of those values. Finally, the element D reflects a device with ports 21, 135 and 445 open, which means that has FTP, MSRPC[2] and Microsoft-DS[3] services available; a probability of 75% of being a 'Windows 2003 Server', or maybe a 'Windows 2000' or an 'XP'; and finally that an indeterminate number of security notes have been described.

	Ports						**W2000**		**XP**	**W2003**	**Security**
	21	**25**	**53**	**80**	**135**	**445**	**SP3**	**SP4**	**SP2**	**Server**	**Notes**
e_1	1	—	—	—	1	1	75%	75%	75%	75%	3
e_2	1	1	1	—	1	1	41%	41%	41%	75%	6
e_3	1	1	—	1	1	1	41%	41%	41%	75%	6
e_4	1	1	—	—	1	1	75%	75%	75%	75%	4
D	21:1	—	—	—	135:1	445:1	W2000 SP3	W2000 SP4	XP SP2	W2003 Servers: 75%	Security Notes

Table 5.3: Description of four elements from *Analia* dataset and their explanation D obtained from AU algorithm.

The new element D can be used to detect regularities between all the elements of a given cluster. These regularities can help security analysts to give them a reason of that clustering result, and to identify possible common problems present in all the elements of that cluster without having to study each element one by one.

[2]MSRPC: Microsoft Remote Procedure Call.
[3]Microsoft-DS: Port used for file sharing in Microsoft Windows.

However sometimes relevant information will never show up due to the inherent behavior of the AU algorithm. This technique only takes into account the features common to all elements. Nonetheless, in certain cases it should be useful to have information not only about common features of all elements but also about features common to most of the elements of the cluster. In the example shown in Table 5.3, security analysts have corroborated that it would be profitable to include in D that the port 25, related to the protocol SMTP, was also open in most of the elements of cluster C. For that reason, the AU algorithm has been modified in order to process *Analia* dataset and give explanations of each cluster. These new explanations include the features common to all elements of a cluster and also incorporate the percentage of occurrence of the attributes not present in all elements. This contribution is presented in the next section.

5.7.2 The Detailed Anti-unification algorithm in Analia

This section details the adaptation of the AU algorithm to consider features not common to all elements but representative enough to be included in the description of the clustering solutions. This new algorithm is named **Detailed AU (DAU)** [24]. In this adaptation, the description of a cluster will include the value of the features common to all the elements in that cluster and common only to some of them, showing the percentage of appearance of those values. A feature will not be included in the description element when the union of the values that the feature takes in the elements is the set of all possible values. After the execution, security analysts will be able to choose the relevance percentage to analyze. The DAU contribution is detailed in Algorithm 5.4.

Algorithm 5.4: Detailed Anti-Unification (DAU) algorithm

```
Let E be the set of elements of a cluster C
Let A a be set of attributes describing the elements of E
Let V_k be the set of all the possible values of the attribute a_k
Let W_ki be the percentage of elements with the attribute a_k holding the value v_i
Let D be the antiunification of the elements of E
Let W_k be the set of all percentages of occurrence of values ∈ V_k of a_k ∈ A
Function DAU is
    forall a_k in A do
        if a_k = v for all e_i in E then
            Store_in_D (a_k, v, W_k =100%)
        Let v_ki be the value that each e_i ∈ E holds in a_k
        if union(v_ki) = V_k then
            a_k will not appear in D
        otherwise
        Let V_k' = union(v_ki) (union of all the values that all e_i in E hold in a_k)
        Let n be the number of elements in E
        forall v_i' in V_k' and v_i' ≠ null do
            W_kj = 100 × (W_kj + 1) ÷ n
            Store_in_D (a_k, v_j', W_kj)
    return D
```

The DAU algorithm is independent of the clustering algorithm used to group elements. Once a clustering technique has been applied to the data set, and elements have been collocated into different clusters, DAU is applied on each cluster to obtain an explanation of the elements gathered together.

An example of DAU application over *Analia* dataset is shown in Table 5.4. Let C_i be a cluster formed by four elements, where e_1, e_2, e_3 and e_4 are network devices whose attributes detail their open ports, possible operating systems and the number of detected security notes. The explanation element D_i of these four devices contains the sharing of common attributes.

	Ports						W2000		XP	W2003	Security
	21	**25**	**53**	**80**	**135**	**445**	**SP3**	**SP4**	**SP2**	**Server**	**Notes**
e_1	1	—	—	—	1	1	75%	75%	75%	75%	3
e_2	1	1	1	—	1	1	41%	41%	41%	75%	6
e_3	1	1	—	1	1	1	41%	41%	41%	75%	6
e_4	1	1	—	—	1	1	75%	75%	75%	75%	4
D_i	21: 1 100%	25: 1 75%	53: 1 25%	80: 1 25%	135: 1 100%	445: 1 100%	W2000 SP3 41-50% 75-50%	W2000 SP4 41-50% 75-50%	XP SP2 41-50% 75-50%	W2003 Servers: 75-100%	Security Notes: 3-25% 4-25% 6-50%

Table 5.4: Description of four elements from *Analia* dataset and their explanation D_i obtained from DAU algorithm.

An example of the summary explanation presented to a security analyst is shown in Figure 5.32. This explanation is built using networking and security vocabulary useful for analysts. In case security analysts want to detect only attributes that are 100% common to all elements, DAU results would be equivalent to AU results. However, DAU allows detecting attributes common to most of the elements of a cluster, information that could also be significant to analysts. For example, in Table 5.4 there is a feature common to most of the elements: port 25, related to the SMTP protocol, is open in the 75% of the devices of that cluster. This feature would have become unnoticeable when applying AU, just because only one element of a cluster does not have this attribute defined, whereas all the other elements of the same cluster share this attribute. Consequently, DAU improves the information given to security analysts as attributes common to the majority of the elements can be also detected.

Description of cluster C_i:
It contains 4 elements.
Information of ports

100% of devices have port 21 open (FTP service open)

100% of devices have port 135 open (MSRPC service open)

100% of devices have port 445 open (Microsoft-DS service open)

75% of devices have port 25 open (SMTP service open)

Information of Operating Systems

100% of devices: Windows 2003 Server in a probability of 75%

Information of Security notes

Different number of security notes

Figure 5.32: Explanation of the elements of Table 5.4 applying DAU algorithm, when the percentage of occurrence of attributes is selected to more than 70%.

5.8 Chapter summary

A main part of any security assessment is the analysis phase. If important information is gathered in the testing phase, but no thorough study is made over this information, the security assessment has no validity and utility. On the other hand, the high volume of collected data demands for the use of automatic techniques to process all the information.

With the *Consensus* framework contribution presented in Chapter 3, security testers can ease the testing burden, but information must still be processed by security analysts. Results from security assessments may notably vary as no two networks behave the same. Then, trying to manually find a behavior pattern for all networks becomes a difficult task. This chapter has presented a new contribution, the improvement of the *Consensus Analysis Module* that includes artificial intelligence techniques to help security analysts in their task of studying data gathered from security tests. This module has been renamed *Analia*. As security analysts do not usually have previous knowledge about the tested network, unsupervised learning techniques may help finding certain device or system patterns. In addition, they may help revealing hidden problems in network security.

An analysis of the machine learning techniques that better fit in the *Consensus* domain has concluded in the use of unsupervised learning. More explicitly, in the application of clustering approaches to group tested devices with similar available services and operating systems and, hence, with similar vulnerabilities. The main phases of the clustering activity have been defined and a contribution for each phase has been presented, designed and included in *Analia*, the *Consensus Analysis Module* with artificial intelligence capabilities.

A discussion regarding the most suitable pattern representations has been included in this chapter. Knowledge representation choice is a crucial factor. The selection of poor or unrepresentative features can deal to incorrect clustering results due to the fact that devices are not symbolized with their main characteristics. In contrast, not all data stored in *Consensus* for each device can be used as input features for clustering algorithms, as not all feature formats are accepted. Then, a thorough study has been presented and the most appropriate features have been selected. A *View Manager* has been designed, implemented and included in *Analia* to configure different pattern representations to be applied over data obtained from security tests in *Consensus*.

Several clustering approaches have been analyzed and a selection of the most used and the most adequate to *Consensus* dataset have been included in *Analia*. In this sense, center-based methods like K-means and X-means have been integrated in *Analia*. A model-based method like Autoclass has been included as well. A soft-computing method based on neural networks, SOM, has been integrated due to its ability to handle uncertain, approximate and complex knowledge. A multiobjective clustering approach, CAOS, has been included to reduce the efforts spent by security analysts. When using this technique, the pursued goals that guide the search for the best solutions are aligned with the validity indices. Thus, the obtained clustering solutions comply with validity requirements. Other clustering algorithms can be easily added to the system, thanks to *Analia*'s modular design. All these clustering approaches are used to cluster tested devices considering features related to operating system fingerprinting, port scanning and vulnerability assessment. The resultant partitions cluster tested devices in groups that share vulnerabilities and threats.

Nevertheless, not only clustering results are important for security analysts. Also the reason of that clustering and explanations that describe why several devices have been grouped together are useful data for them. This is why another contribution of this thesis that has been integrated in *Analia* is related to the data abstraction phase. First, an adaptation of the Anti-unification algorithm has been performed to explain clustering results to analysts by using the same lexicon that security analysts are familiarized. Analysts are not used to centroids or director vectors that

can justify a clustering solution from the machine learning point of view. But a description of a clustering result can help them in their task. Therefore, the AU algorithm has been adapted to the *Consensus* dataset. Afterwards, several improvements have been proposed for the AU algorithm. Finally, a new algorithm called Detailed Anti-unification algorithm (DAU) has been presented as a contribution and has been integrated in *Analia*. DAU improves the explanations given to security analysts as not only attributes common to all elements, but also other attributes common to the majority of the elements, are included in the cluster descriptions.

The cluster validity phase has been analyzed in order to include it in *Analia* and contribute with new validity indices, designed *ad-hoc* to this security environment. The most used validity indices have been analyzed and integrated in *Analia*. Dunn, Davies-Boudin and Silhouette indices can be used to evaluate clustering results and select the best partitions. Besides, new validity indices, the Intracohesion and Intercohesion factors, have been presented to evaluate clustering results. They are based not only on the input features used to cluster tested devices, but also on other important features stored in *Consensus* database that characterize those clustered devices. All these indices are included in *Analia* to predict the best number of clusters in clustering approaches that need the configuration of this parameter beforehand.

Finally, *Analia* integrates a whole clustering process to be used in the *Consensus* framework, with data gathered from exhaustive security assessments. This clustering process has been designed modularly so it is not necessary to apply all the phases if not required. But the whole set of functions constitute a unique system that allows the clustering of stored information previously collected from security tests. So *Consensus* along with *Analia* constitute the core contribution of this thesis.

Chapter 6

Analia: Experimentation

The experimentation contributes to the better understanding of the processes related to a security audit that have been included in the analysis module of *Consensus*, named *Analia*. This experimentation also demonstrates that security analyst requirements are covered with the use of this new global contribution. This chapter summarizes the experimentation that we have carried out with *Analia* to demonstrate its usefulness in the security domain. Data gathered in *Consensus* has been processed with *Analia* and we have made full use of its different features to extract conclusions from the information of security vulnerability assessments. The different experiments are described and experimentation results are detailed. These results corroborate how *Analia* can be used to improve the daily work of a security analyst.

6.1 Introduction

The last phase of a security assessment implies an structured analysis of the collected data and a reporting approach [97]. *Consensus* gathers data from different security testing tools that may produce incoherent and scattered outputs for a same audit. Moreover, an assessment of a network with a high number of devices may generate a large amount of data. Afterwards this information must be processed. In that respect, the *Analia* system has been presented as a main contribution to address several issues related to the analysis phase. *Analia* includes AI techniques to improve this final stage.

This chapter outlines the structure of the empirical study performed with *Analia*, the analysis system detailed in Chapter 5. It provides a description of the different phases of the experiments, the scenarios of this experimentation, the machine learning techniques that have been tested and the analysis of the results.

Different testing scenarios were previously described in Chapter 4. This previous experimentation was done not only to check *Consensus* performance but also to collect many testing data to be further analyzed. The data used as starting point has been extracted from the preceding experiments carried out on La Salle - URL data network. The characteristics of this network are known beforehand. This fact facilitates the interpretation of the analysis results.

First experiments were performed out to analyze the feasibility of applying machine learning, and more specifically, unsupervised learning techniques to data stored in *Consensus*. This is why some of the most well-known clustering algorithms were studied and applied to that dataset. K-means [60], X-means [96], *Autoclass* [21] and SOM [72] were the chosen algorithms, as explained in Section 5.2. The positive results of this experimentation encouraged us to keep on working on this area, providing a contribution to improve the validation of clustering results. This experimentation is also described in this chapter. The different results obtained from the various clustering approaches forced us to define a procedure to select the best clustering solution for a given dataset. This step was improved with the inclusion of the multiobjective clustering technique based on evo-

lutionary algorithms called CAOS [48] and these experiments are detailed as well [27]. Finally, an enhanced way to show the analysis results to security experts was required. The final goal is to help security analysts, so special knowledge of AI is not needed to use Analia. In this sense, results displayed by *Analia* should be understandable and helpful in the security context. This is why experiments have been executed with the aim of improving the stage of result explanations. These tests are also explained in this chapter. In the next sections all this experimentation and results are detailed.

6.2 Initial experimentation with Partition Methods

This initial experimentation was conducted to demonstrate the viability of applying clustering techniques to the data gathered from security assessments that was stored in *Consensus*. In addition, the different knowledge representations described in Section 5.4 were analyzed.

The dataset used for this experimentation was extracted from the security tests done over La Salle network (see Section 4.3). From the first security tests, 34 distinct devices were selected: 16 from three different student labs (5+2+9), 8 from the Networking laboratory testbed, and 10 public servers. Three pattern representations were used to handle *Consensus* data. All these three pattern representations included information regarding port scanning and operating system fingerprinting. These pattern representations have been previously discussed in Section 5.4.4 and were implemented in three different views. They are the following:

- **View 0**: List of all available ports and OS. This representation includes more than fifty features related to open ports and eleven features related to possible OS (see Figure 5.13).
- **View 1**: This representation includes a single port feature with the sum of the open ports, and the list of all OS available (see Figure 5.14).
- **View 2**: This representation is a multidimensional vector where information regarding open ports is combined in ranges, and the value of each feature corresponds to the sum of the open ports in the appropriate range. The ranges are the following: 1-400, 401-999, 1000-10000, 10001-65535. Ranges are smaller in the lower ports, due to the fact that these are the ranges where the majority number of ports are open. So a better granularity is allowed in the lower scope. The other features are the list of all OS available (see Figure 5.15).

K-means was the first unsupervised learning technique used to cluster the data stored in *Consensus*. This algorithm requires specification of the number of clusters to get (K), which was an unknown value in this environment. This is why several executions were run, varying not only K but also the value of the initial seed. Tests have been executed changing K from 2 to 9 and using 20 different seeds. The initial dataset was represented by the three views detailed before.

Results of this experimentation are shown in Figure 6.1. The first column shows the identifier of each device. View 0 is the knowledge representation used as input. Clusters are assigned different frame styles. When dividing the set into two clusters (K=2), all the *Windows* devices have been grouped together, independently of their OS version or their open ports. The other cluster has the non-Windows devices, like Linux or Solaris servers. For K=3, the Windows cluster remains, whereas the non-Windows cluster has been divided into two new clusters: one cluster includes all the public servers except for one; the other cluster comprises 4 devices from the testbed and a public server. For K=4, the *Windows* cluster remains, there is a cluster of 2 public servers, a cluster with 4 devices from the testbed, and a cluster with 8 public servers. In this configuration, the public servers are divided into two groups: the cluster with two devices comprises 2 Linux servers that share many services, whereas the cluster with 8 devices includes the Solaris servers and the rest of

the Linux servers with different open services. For $K>4$, an important fact is detected: a cluster with device 16 splits off. This is a Windows 2003 computer of a student laboratory and it should have been clustered with the rest of the lab machines. The devices of that lab should share the same properties, due to the fact that the CSI manages these devices and installs the same image for all of them. However, device 16 has been manipulated and its properties have been modified. Therefore, the clustering approach separates it in a different cluster. This result corroborates that *Analia* helps detecting rogue or modified devices from a security assessment without having to analyze the whole dataset. Security analysts may detect this fake device very easily with the aid of *Analia* and countermeasures can be applied immediately to solve the problem.

This experimentation shows that a new group of two Linux devices of the testbed is created for $K=6$. In fact, the testbed has Linux and Windows devices with different features to check the operation of *Analia*. So these devices become to be grouped in smaller clusters when increasing the values of K. Even more, for $K\geq 7$, the device 0 is a single member of a cluster due to the different services that this testbed device offers, and most of these services are not shared with the rest of the testbed devices. The Solaris devices (24, 27 and 30) with specific services also split off a new cluster for $K\geq 7$. The Windows devices are part of the same cluster until $K=9$, except for device 16, fact already discussed. For 9 clusters, three Windows clusters are formed: one with the Windows 2003 devices, one with the Windows NT devices, and one with the modified device. This distinction between Windows versions had not been detected with less clusters because the common open services predominated. These three clusters still allow analysts to corroborate that only device 16 was substantially changed, so a new cluster was created for this device.

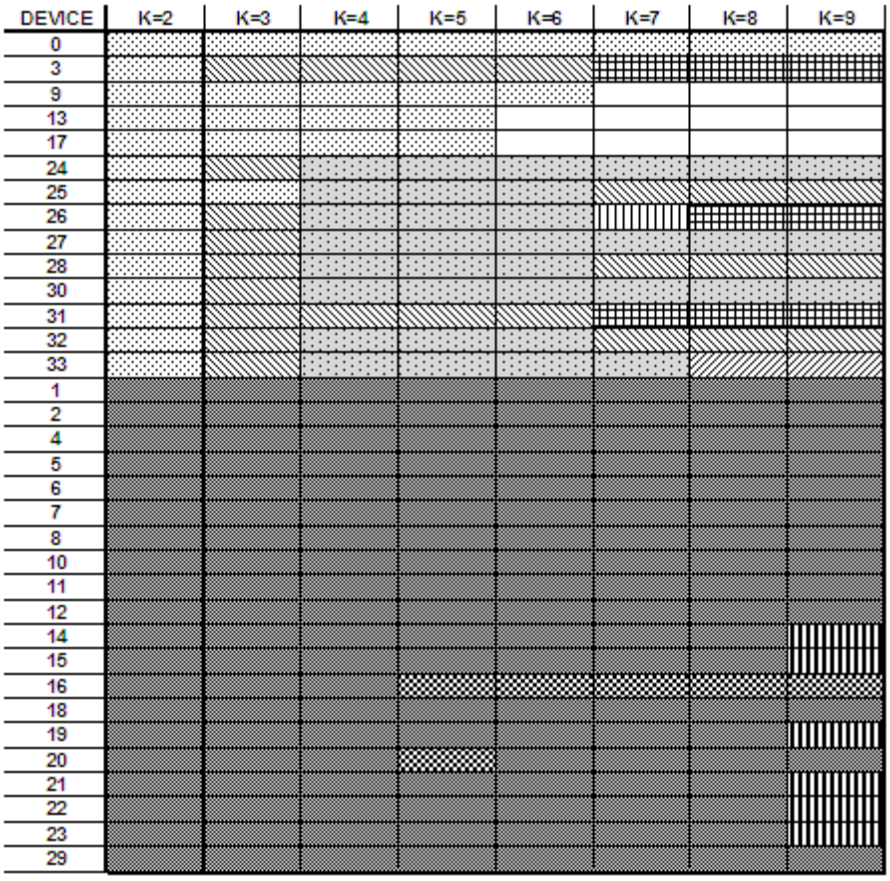


Figure 6.1: Cluster results with K-means for different values of K.

Another way to display clustering results was studied. This visualization represented clusters by clouds, as Figure 6.2 depicts. However this display was discarded due to the fact that this representation is completely subjective and only shows changes between executions with different values of K. The variations between configurations can be graphically detected, but this was not what the security analyst expected. The final desired output is a list of the different clusters and their devices associated. An example of this output is shown in Figure 6.3. It shows the results of a K-means execution with K=5, including the IP addresses of the devices contained in each cluster. *Analia* keeps a connection between each device identifier and the respective IP address with the rest of the security data.

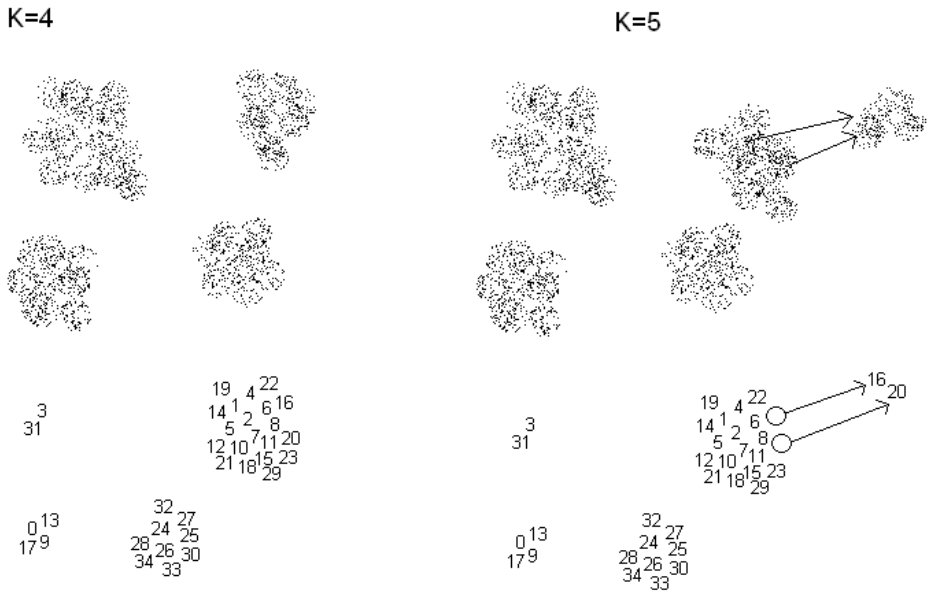


Figure 6.2: Display of cluster results with K-means for K = 4, 5.

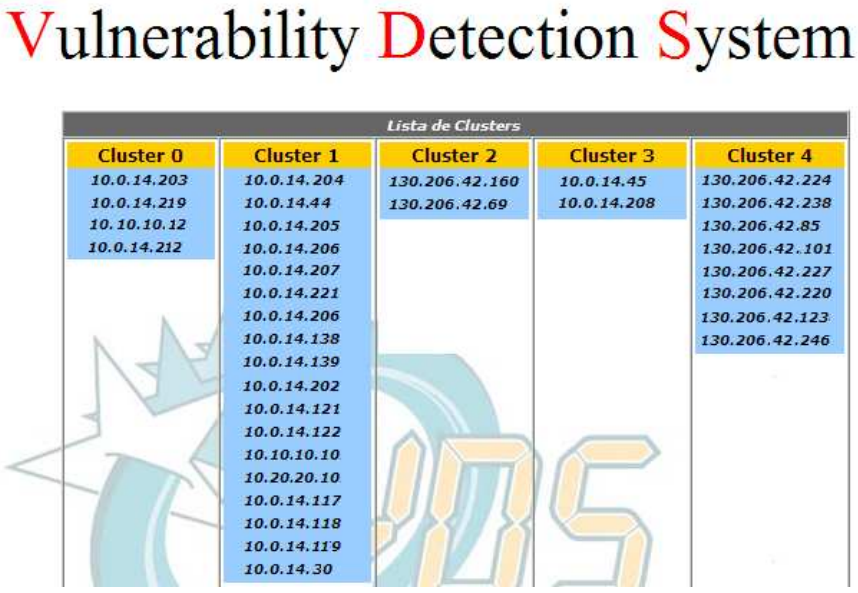


Figure 6.3: Display of cluster results with K-means for K = 5 in *Analia*.

These first experiments showed that clustering techniques can be very useful not only identifying groups of similar computers but also finding computers that unexpectedly appear separated with different characteristics. Thus *Analia* aids analysts detecting suspicious devices in a controlled environment, where all computers are expected to behave the same way, and finding any anomaly could be of a great usefulness [28].

A shortcoming of K-means is the need to specify initially the number of clusters. In this unsupervised domain, this is an unknown value. So several executions with different values of K had to be performed in order to select the best results. On the other hand, an algorithm like X-means is capable of calculating automatically the best number of clusters. This is why different tests were executed with the same dataset using K-means and X-means approaches.

X-means has the added difficulty to automatically calculate the number of classes. Nonetheless, similar results have been obtained in X-means executions comparing to K-means results, where K has been tuned manually to get the best classification. These results are shown in Figure 6.4. The application of these techniques was considerably new in this network security data, so the resulting clusters had to be interpreted and further analyzed to extract the final conclusions [28]. The methods had grouped all devices' information into clusters, but the distribution of the computers within the clusters seamed unconnected. It was not until the devices' real features were directly visualized that results started to be acknowledgeable.

K-means

FILE 1		FILE 2		FILE 3	
PC	OS	PC	OS	PC	OS
0	Linux	0	Linux	0	Linux
		9	Linux	13	Linux
9	Linux	13	Linux	17	Linux
13	Linux	17	Linux		
17	Linux			3	Linux
		3	Linux	9	Linux
3	Linux	25	Linux		
26	Linux	26	Linux	25	Linux
31	Linux	28	Linux	26	Linux
		31	Linux	28	Linux
33	Linux	32	Linux	31	Linux
		33	Linux	32	Linux
25	Linux			33	Linux
28	Linux	24	Solaris		
32	Linux	27	Solaris	24	Solaris
		30	Solaris	27	Solaris
24	Solaris			30	Solaris
27	Solaris	19	NT		
30	Solaris	22	NT	14	NT
		29	NT	15	NT
14	NT			19	NT
15	NT	21	NT	21	NT
19	NT	23	NT	22	NT
21	NT			23	NT
22	NT	14	NT		
23	NT	15	NT	1	XP
29	NT			2	XP
		1	XP	4	XP
1	XP	2	XP	5	XP
2	XP	4	XP	6	XP
4	XP	5	XP	7	XP
5	XP	6	XP	10	XP
6	XP	7	XP	11	XP
7	XP	8	XP	12	XP
8	XP	12	XP	20	XP
10	XP	18	XP		
11	XP	20	XP	8	2003
12	XP			18	2003
18	XP	10	XP		
20	XP	11	XP	16	XP
16	XP	16	XP		

X-means

FILE 1		FILE 2		FILE 3	
PC	OS	PC	OS	PC	OS
0	Linux	0	Linux	0	Linux
9	Linux	9	Linux	9	Linux
13	Linux	13	Linux	13	Linux
17	Linux	17	Linux		
32	Linux			3	Linux
		3	Linux	28	Linux
3	Linux	25	Linux	31	Linux
26	Linux	26	Linux	32	Linux
28	Linux	28	Linux	33	Linux
31	Linux	31	Linux		
		32	Linux	17	Linux
25	Linux	33	Linux	25	Linux
33	Linux				
		24	Solaris	26	Linux
24	Solaris	27	Solaris	30	Linux
27	Solaris	30	Solaris		
30	Solaris			24	Solaris
		19	NT	27	Solaris
14	NT	21	NT	30	Solaris
15	NT	22	NT		
19	NT	23	NT	14	NT
21	NT	29	NT	15	NT
22	NT			19	NT
23	NT	14	NT	21	NT
29	NT	15	NT	22	NT
				23	NT
10	XP	10	XP		
11	XP	11	XP	10	XP
				11	XP
1	XP	1	XP		
2	XP	2	XP	1	XP
4	XP	4	XP	2	XP
6	XP	6	XP	4	XP
7	XP	7	XP	5	XP
12	XP	12	XP	6	XP
				7	XP
8	2003	8	2003	12	XP
18	2003	18	2003	16	XP
		20	2003	20	XP
5	XP				
		5	XP	8	2003
16	XP			18	2003
20	XP	16	XP		

Figure 6.4: Distribution of clusters by K-means and by X-means.

Experimentation was carried out with the three different views explained before. The tables shown in Figure 6.4 depict the output results of K-means and X-means executions. For the same input dataset, the column named *File 1* corresponds to the View 0, the column *File 2* corresponds to the View 1, and the column *File 3* corresponds to the View 2. Results show that the network devices were grouped by their OS and port features. Regarding the View 0, it can be seen that K-means has clustered device 16 into a separated cluster, even theoretically belonging to the Windows XP group. However, the same device is grouped with device 20 in the X-means solution, because both have a very similar port map and the same OS. Regarding the View 1, both techniques obtained a much more compact cluster grouping. Again, device 16 was clustered separately. Regarding the View 2, some clusters became even more compacted and the rest were empty, due to the fact that this knowledge representation used the same number of port attributes than OS. In K-means the device 16 appeared again separated from the others, whereas in X-means it was merged with the rest of the XP devices.

Both techniques have revealed good results when using port scanning and OS fingerprinting information. They have identified groups of similar computers. Moreover, they have also discovered a device that unexpectedly was separated from what it seamed like similar devices. Thus, these techniques can help analysts handling information obtained from security tests to detect abnormal groups of devices or atypical system behaviors. In addition, security analysts asserted that implementing such methods in network security testing could aid assessing test results [28].

However several issues needed to be addressed. This initial experimentation helped us to corroborate the convenience of merging machine learning techniques in a system like *Consensus* and, more specifically, in its analysis module called *Analia*. Then, more unsupervised learning techniques had to be analyzed to determine its suitability in this security environment. Also the selection of the best execution to offer security analysts the best clustering results had to be studied. Furthermore, the possibility of explaining to analysts the reason of those solutions needed a further analysis. All these questions are handled in the next sections, where the different contributions presented in *Analia* have been experimented.

6.3 Experimentation with Autoclass

This section describes the experiments performed with Autoclass [21] over *Consensus* dataset. The Autoclass results are compared with the K-means solution for the same input data. An advantage of Autoclass compared to K-means is that the former can calculate automatically the number of clusters but its performance should be evaluated as well.

Experimentation has been run for automatic and manual Autoclass, where the number of classes has been set from 2 to 10. Autoclass requires specification of the maximum number of iterations, which has been configured with the following values: 10, 50, 100, 200, 500 and 1000.

The dataset to classify comes from the data gathered from La Salle network (see Section 4.3). It included data from 44 tested devices: 21 (14+7) from two different student labs, 9 public servers, 11 internal servers and 3 staff computers, all with different OS and open ports. It has been represented with three different knowledge patterns (see Section 5.4.4), which are the following:

- **View 0**: List of all available ports and operating systems (see Figure 5.13).
- **View 1**: List of all OS available and a single feature with the sum of the open ports (see Figure 5.14).
- **View 2**: List of ranges of open ports; the value of each feature is the sum of the open ports in the appropriate range. The ranges are the following: 1-400, 401-999, 1000-10000, 10001-65535. The other features are the list of all OS available (see Figure 5.15).

The clustering results of the best executions of automatic Autoclass for the different views are shown in Figure 6.5. For the View 0, automatic Autoclass has divided the input dataset into 5 clusters: a cluster with all the 14 devices of a student lab; a cluster with 5 Windows devices of the other student lab, 8 Linux internal servers and 2 Linux external servers; a cluster with 4 Linux public servers and 2 Solaris public servers, a Linux internal server and a staff computer; a cluster with a Linux public server, 2 lab devices and a staff computer; and a cluster with 2 Windows internal servers and a staff computer. When asked to security analysts, this clustering solution looked like strange for them. The second cluster grouped many diverse devices with very different operating systems (Windows and Linux) and with many different open ports. As an example, the lab devices had nothing in common with the public e-mail server but they were clustered together.

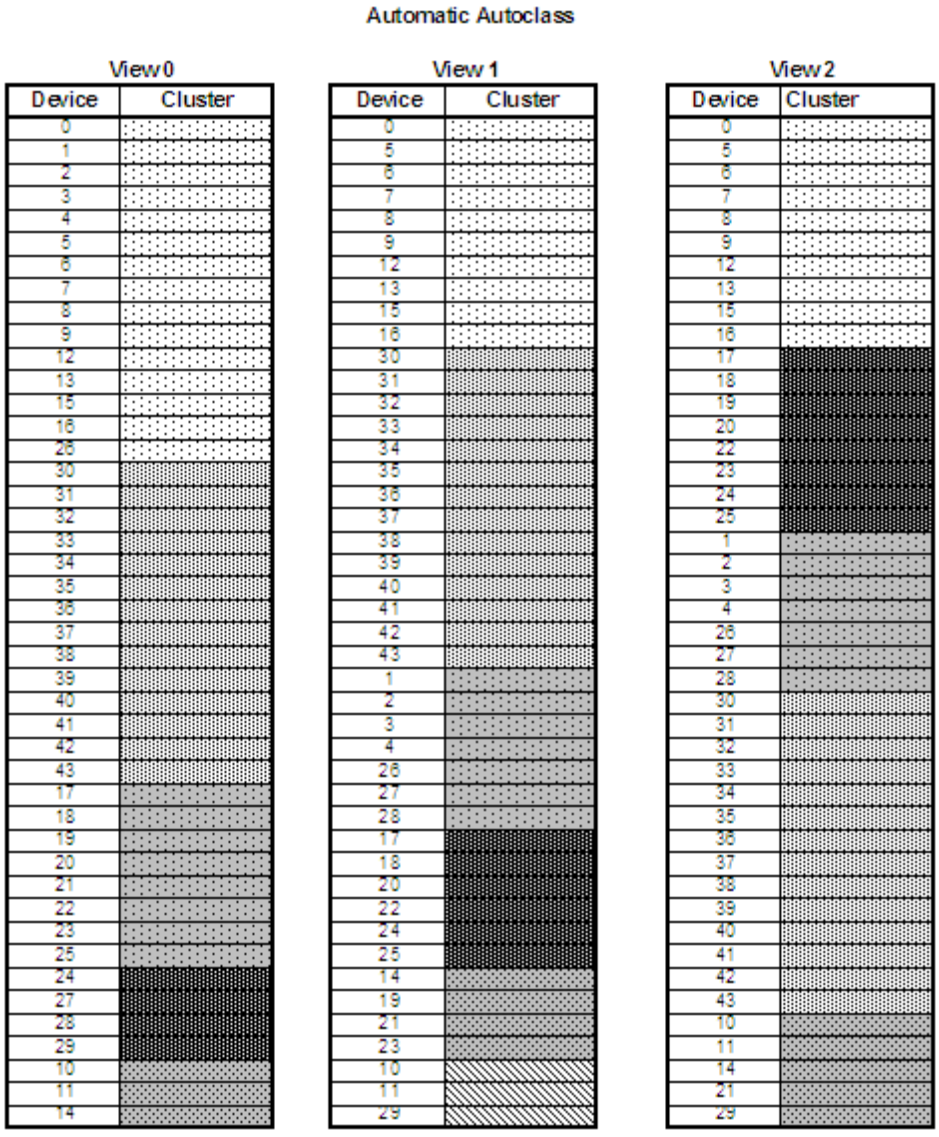


Figure 6.5: Cluster distributions by automatic Autoclass for Views 0, 1 and 2.

Regarding the View 1, automatic Autoclass has divided the input dataset into 6 clusters: two clusters with the student devices, separated by their lab, that is to say, 14 devices in a cluster and 7 devices in the other cluster; a cluster with 8 internal and 2 external servers, all Linux with similar open services; a cluster with 5 Linux external servers and an internal server; a heterogeneous cluster

with two external Solaris servers and 2 staff devices; finally, a cluster with two internal servers and a staff computer. This striking clustering of Windows and Solaris devices without common services did not convince security experts.

Regarding the View 2, automatic Autoclass has divided the input dataset into 5 clusters. Three clusters were the same ones obtained from the View 1: 2 clusters with lab devices from two different labs, and a Linux cluster with 8 internal and 2 external servers. A new cluster was created with two external Solaris servers, 5 Linux external servers and a Linux internal server that shared several services with the rest of the devices. The last cluster included the 3 Windows staff devices and two Windows internal servers. This configuration has prioritized operating systems with respect to open services, although the grouped devices shared some services as well.

Clustering with automatic Autoclass shows an improvement in the calculation of the most appropriate number of clusters. However the solutions provided by this experimentation did not help security analysts to a large extent, due to the fact that some solutions grouped very dispar operating systems with no common services. Manual Autoclass showed similar clustering results, with the further difficulty that the number of clusters had to be configured previously. Thus security analysts preferred K-means to Autoclass results. This qualitative discussion is reinforced by the analytic study presented in Section 6.5.

6.4 Experimentation with SOM

This section details the experimentation done with SOM [72] over *Consensus* dataset. The selection of this unsupervised learning technique was motivated by it soft computing capabilities that make SOM suitable to work with data complex problems [47].

This experimentation was performed with the data gathered from La Salle network (see Section 4.3). This dataset included data from 44 tested devices: 21 (14+7) from two different student labs, 9 public servers, 11 internal servers and 3 staff computers. These devices had different open ports and implemented very diverse operating systems. The view used to represent the knowledge was the View 0. It listed all available ports and operating systems (see Figure 5.13). This input file contained more than 160 features. This is because a wide variety of devices had been tested to get as much representation as possible. This view was used as input file for the K-means, X-means and SOM approaches. All algorithms have been executed to test different values of K for K-means and different maps of SOM. Besides, different initial random seeds have been configured to check their influence.

The three approaches clustered the dataset into different groups. An example of clustering results for K-means and the View 0 is shown in Figure 6.6. For K=2, K-means divided the tested devices into two clusters: the Windows devices and the non-Windows devices. For K=3, the cluster with Windows devices remained intact, whereas a new cluster with a single element, device 0, split off from the non-Windows cluster. This device was the public e-mail server, that was a Linux server with other specific services different from the other devices. For K=4, the two non-Windows clusters were maintained, whereas two clusters with Windows devices were created: one with the 7 devices of an student lab; another with the rest of the Windows devices. For K=5, the two Windows clusters and the cluster with device 0 remained equal. Two new clusters were created: one with the Linux servers that hosted many common applications for other institutions bounded to La Salle - URL, and another cluster with the Linux and Solaris servers with the services hosted for the university users (students and staff). For K=6, the three clusters for non-Windows devices continued the same, as well as the cluster with the 7 devices of a student lab. A new cluster was created with the 14 devices of the other student lab. The last Windows cluster included the 3 staff computers and 2 internal servers. K-means was not able to calculate more than 6 clusters, providing empty clusters for configurations of K>6.

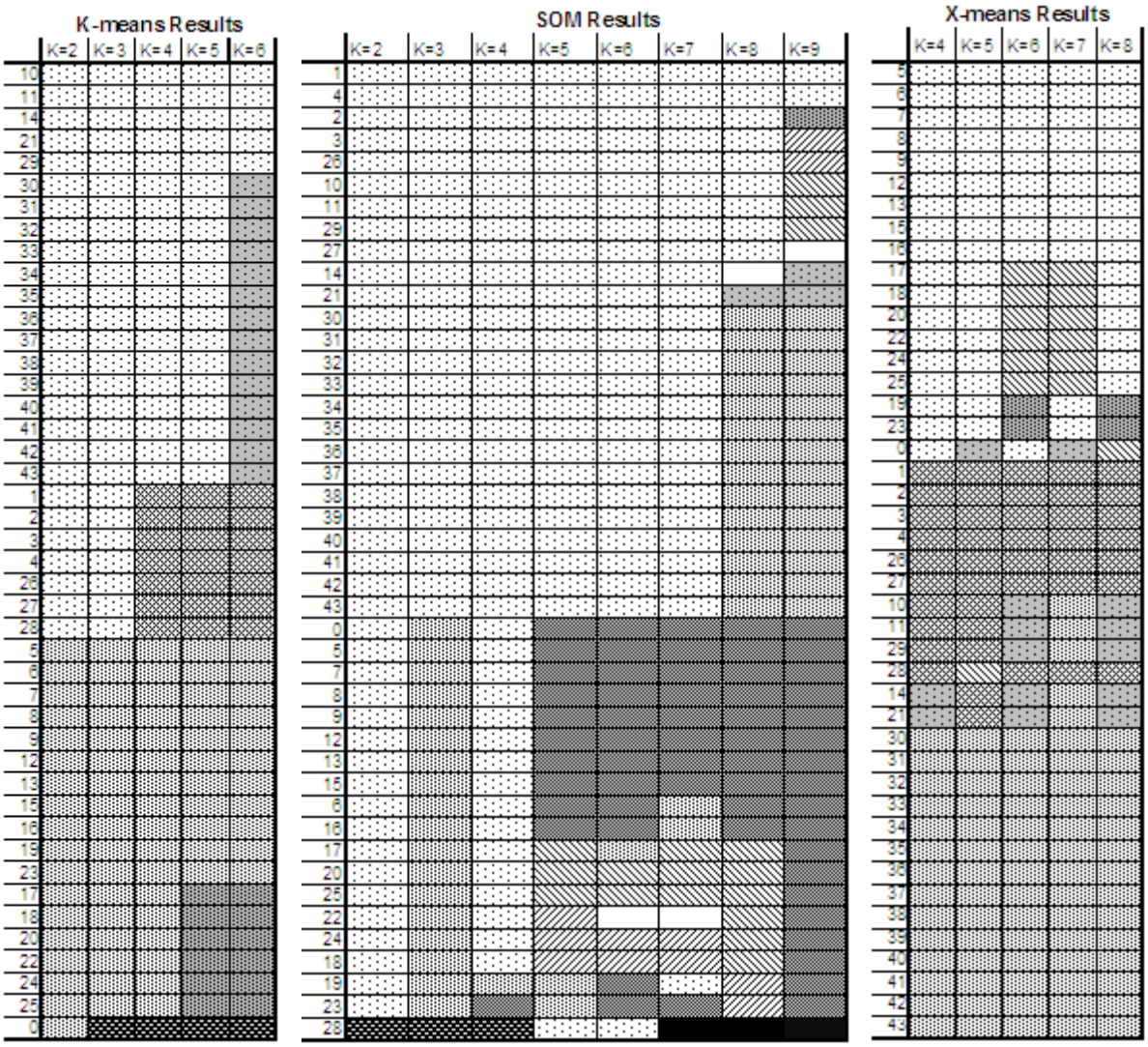


Figure 6.6: Distribution of clusters by K-means, X-means and SOM for View 0, where K is the number of clusters.

SOM has been tested using several map sizes of 2-dimensions (from 4 clusters $-2 \times 2-$ to 36 clusters $-6 \times 6-$ to analyze several data dispersions), two different distance measures (normalized Euclidean and normalized Hamming distance) and 10 random seeds (to minimize the random effects of initialization). The learning factor, the neighbor factor and the maximum number of iterations have been set to typical values: from 0.6 to 0.01, from M to 1 and 500 iterations by neuron.

An example of clustering results for SOM and the View 0 is shown in Figure 6.6. SOM clustered devices in a different way compared to K-means results. For K=2, SOM did not divided the space into Windows or non-Windows devices. Instead of that, a cluster with a single element was created (device 28). This device was a student lab computer with many filtered ports, which made it singular with respect to the rest of the devices. This is why this device was clustered alone in most of the SOM solutions (see Figure 6.6). The other cluster grouped the rest of the devices. This configuration allows concluding that a computer from a student lab was altered, as it should have been grouped with the rest of the computers of the same lab. In contrast, K-means was

not able to distinguish this fact. For K=3, SOM divided the scope into 3 clusters: the Windows devices, the non-Windows devices and the device 28, the one with filtered ports. This configuration was very similar to K-means solution for K=3, except for the single device, where each approach stood out for different device properties. However for K=4, SOM clustered together Windows and non-Windows devices and produced three other clusters with a single element. One of these clusters contained the device 28, and the other two clusters separated the two Solaris public servers (devices 19 and 23, respectively). This configuration was useful to detect three singular elements, but the cluster with the rest of the elements was a bit of a mixed bag. For K=5, all the Windows devices except for 28 where grouped together. The 8 Linux internal servers and 2 Linux external servers with similar open ports were clustered. The two relay servers and the other Linux internal server shared many services, so they were grouped. The other three public servers were separated into another cluster. Finally, there was another cluster with the two Solaris servers. For K=6 and 7, the Windows clusters were maintained and the differences were found in the *Linux* part. The public web server was placed in a new cluster (device 22). A cluster with two identical internal servers (devices 6 and 16) was also created for K=7. Finally, for K=8 and 9, a new cluster with the 14 computers of a student lab was created, as in K-means with K=6. Other clusters were repeated from previous solutions, like the Solaris cluster, or the lab computer with filtered ports. Windows devices were separated depending on their staff function (devices 14, 21) or their internal server function. The final configuration divided the student lab of 7 devices into 4 different clusters, due to the fact that some of the computers had been slightly modified but these subtle changes had gone unnoticed until this final configuration.

An example of clustering results for X-means and the View 0 is shown in Figure 6.6. X-means calculates the ideal number of clusters automatically given a range of possible values. So different ranges were configured to obtain different results. The 14 devices of a student lab were always clustered together, including other 2 staff computers. For K=4, all non-Windows devices were grouped, a cluster for 2 staff devices was created, and a cluster with the 7 lab devices, 2 Windows internal servers and a staff computer was built. For K=5, the public e-mail server was separated from the other non-Windows devices. As the SOM solution, X-means also detected changes in device 28 due to the fact that it was separated from the rest of the devices of the same lab. For K=6, the Solaris devices formed a new group (19 and 23); public and internal servers were divided into two clusters. However, this configuration clustered again all the 7 lab devices, masking the discovery of the computer modified. Configurations with higher values of K returned empty clusters, as it can be seen in Figure 6.6. Therefore, X-means was not able to return solutions with more than 6 clusters, like K-means.

In addition, network security analyst experts analyzed the results obtained after clustering the data set. For example, for K=4, human experts observed that K-means and X-means grouped devices prioritizing operating systems. However, SOM grouped devices that had similar vulnerabilities although they implemented different operating systems, as SOM is more appropriate to deal with complex knowledge. All clustering proposals were very interesting for them, as they can help analysts handling information obtained from security tests to detect devices with similar vulnerabilities, abnormal groups of devices or atypical system behaviors [26]. But a problem arises here. Which clustering solution should be presented to security analysts? They do not need to be shown many clustering solutions. They need a system that makes work much easier but not much harder. With this analysis, the best solution could be selected from a qualitative point of view. However, this process could no be automated. The goal of *Analia* is to alleviate analysis work, so the system needed a mechanism to quantitatively select the best clustering solution given a set of executions. This is why a Student's t test was first carried out and, afterwards, clustering validation techniques were also included in the system. This experimentation is detailed in the next sections.

6.5 Experimentation with cluster validation

Clustering allows dividing the dataset of tested devices into partitions that group elements that share operating systems or open services. However the number of partitions is not known a priori. For that reason the real challenge is to be able to evolve a proper value of the number of clusters and provide the appropriate clustering solution. Thus a validity method must be applied to verify clusters quality. Moreover, the same validation method can be used to calculate the best number of clusters for the given problem [28]. The experimentation performed regarding cluster validation techniques is detailed below.

The dataset used to apply the validation techniques was composed of the same 44 devices explained in Section 6.4. The *Cohesion factors* described in Section 5.6 as a contribution of this thesis were applied to evaluate the goodness of the different clustering solutions [26]. These values were calculated for different partitions of the same dataset to check the influence of the K value or the size map variation. Ten different random seed values were configured to minimize the effects of initialization. The main goal was to corroborate that clustering approaches grouped elements not only by their common open services and operating systems but also by their common security vulnerabilities. Cohesion factors were designed to evaluate how similar devices are in a cluster and how distinct devices are in different clusters in terms of security vulnerabilities. This is why the experiments detailed in this section are related to Cohesion factors, although other experiments have been run with the typical indices for clustering validation: Dunn [40], DB [33] and Silhouette [103].

The values shown in Table 6.1 represent the mean of the mean Intracohesion factor for every execution and its standard deviation, the mean of the mean Intercohesion factor for every execution and its standard deviation, and finally the difference between the means of the mean Intracohesion and mean Intercohesion. Although K-means algorithm was configured with K from 2 to 10 and different seed values, this approach was not able to distribute the elements in more than six clusters, as explained in Section 6.4. On the other hand, SOM was able to create more than six clusters in several map size configurations. SOM was configured with size maps of 2×2, 3×3, 4×4, 5×5 and 6×6 (see Section 6.4).

As stated before, the most interesting values should include a high mean Intracohesion and a low mean Intercohesion. If Intracohesion has a high value, the elements of a cluster are very similar to the other elements of the same cluster for every cluster of the system. This fact implies that the devices of the same cluster have many security vulnerabilities in common, so the same countermeasures can be applied to all the members of a cluster to solve those vulnerabilities. If Intercohesion has a low value, clusters are very different between them. This statement implies that devices from two different clusters share few vulnerabilities or even they do not share common vulnerabilities.

Table 6.1: Summary of the mean of Intracohesion and Intercohesion values for K-means and SOM algorithms using different number of clusters (C). It also includes the difference between both factors. The best results are marked in **Bold**.

Clusters (C)	***K*-means**			**SOM**		
	Intrach. (std.)	Interch. (std.)	Diff.	Intrach. (std.)	Interch. (std.)	Diff.
3	0.661 (0.09)	0.183 (0.08)	0.477	0.680 (0.07)	0.095 (0.13)	0.584
4	0.687 (0.06)	**0.161 (0.03)**	**0.52**	**0.875 (0.01)**	**0.088 (0.01)**	**0.786**
5	0.693 (0.04)	0.216 (0.02)	0.477	0.678 (0.07)	0.231 (0.12)	0.447
6	**0.706 (0.01)**	0.217 (0.01)	0.488	0.748 (0.08)	0.288 (0.08)	0.459
7	—	—	—	0.762 (0.04)	0.318 (0.04)	0.443
8	—	—	—	0.817 (0.03)	0.355 (0.09)	0.461

Considering only the mean Intracohesion value, it can be seen that, in general, it increases when the number of clusters increases. The reason is the following: when having more clusters, the cluster size decreases as there are more groups to insert the devices; consequently, it is easier that the elements in a cluster look like more similar. Conversely, in general when the number of clusters increases the mean Intercohesion also increases, which is not the desired result. The more number of clusters, the higher possibility that two clusters become more similar because maybe the differences between devices are not so significant to create so many clusters. As a result, it is necessary to find a compromise between both cohesion metrics and this compromise will fix the best number of clusters for this dataset. Another value that can be useful to help deciding the best solution is the difference between the mean Intracohesion value and the mean Intercohesion value. A higher value is good and the best values are in bold in Table 6.1.

Figure 6.7 shows the evolution of Intracohesion and Intercohesion factors comparing both clustering approaches. The best solution for SOM Intracohesion consists of four clusters, whereas K-means considers six clusters, although other solutions have similar Intracohesion rate. Regarding Intercohesion, both algorithms agree in the best configuration with C= 4. Although K-means obtained a better Intracohesion for C=6, its Intercohesion was lower. In conclusion, the best number of clusters for this data set is 4, considering the values of Cohesion factors. Furthermore, SOM has calculated the best configuration [26].

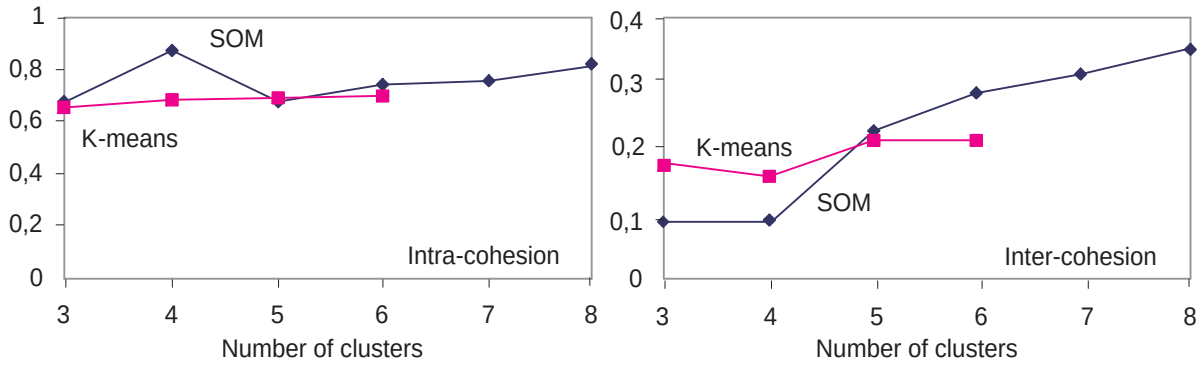


Figure 6.7: Intracohesion and Intercohesion evolution for K-means (■) and SOM (♦).

For the *X-mean* solutions showed in Figure 6.6, the best Intracohesion value was for the solution of 5 clusters, whereas the best Intercohesion value again was for the solution of 4 clusters. The difference between both values also got better results for C=4. Therefore, for this dataset the best partition was considered of four clusters.

A Student's t test was run to compare Cohesion factor results between SOM and K-means approaches. The confidence interval was set to 95%. Results are shown in Table 6.2. Regarding the Intracohesion factor, SOM results were significantly better than K-means results for 4 and 6 clusters. Regarding the Intercohesion factor, SOM results were significantly better for 3 and 4 clusters. Conversely, K-means results were significantly better for 6 clusters. There was no significant difference between results of both techniques for 5 clusters. In conclusion, for the best partition C=4, SOM performed significantly better than K-means in both Cohesion indices.

Number of clusters	**3**	**4**	**5**	**6**
Intracohesion factor	⇌	↑	⇌	↑
Intercohesion factor	↑	↑	⇌	↓

Table 6.2: Student's test result of SOM compared to K-means.

In addition, a comparative study has been performed between K-means and Autoclass to side with one of the approaches. The dataset used to apply the validation techniques was composed of the same 44 devices explained before. The validity factors detailed in Section 5.6 where used to evaluate Autoclass response. Autoclass was executed in automatic mode and in manual mode, tuning the number of clusters from 2 to 10. The maximum number of tries was parametrized to 10, 50, 100, 200, 500 and 1000, as detailed in Section 6.3.

Figure 6.8 shows the Cohesion factor values for K-means and Autoclass approaches. Intracohesion and Intercohesion factors are calculated as the mean of the executions for each class obtained. For the Intracohesion value, the K-means graphic in Figure 6.8 has its maximum for 6 classes. After running a Student's t test, the system shows that there is not a significant difference of results of 4 classes amongst results of 3 or 5 classes. This test also states that the solution with 4 classes is significantly different from the solution with 6 classes. Regarding Autoclass, both versions prefer 5 classes. However, results of automatic Autoclass for 5 classes are not significantly different from the 6 classes solution. For the Intercohesion value, none of the algorithms agree in the best output. K-means selects 4 classes as detailed before, automatic Autoclass returns 6 classes and Autoclass with a fixed K returns 3 classes as the best solutions. Regarding the difference between Cohesion factors, K-means selects 4 classes as the best solution, while Autoclass coincides with the same results of the Intercohesion factor.

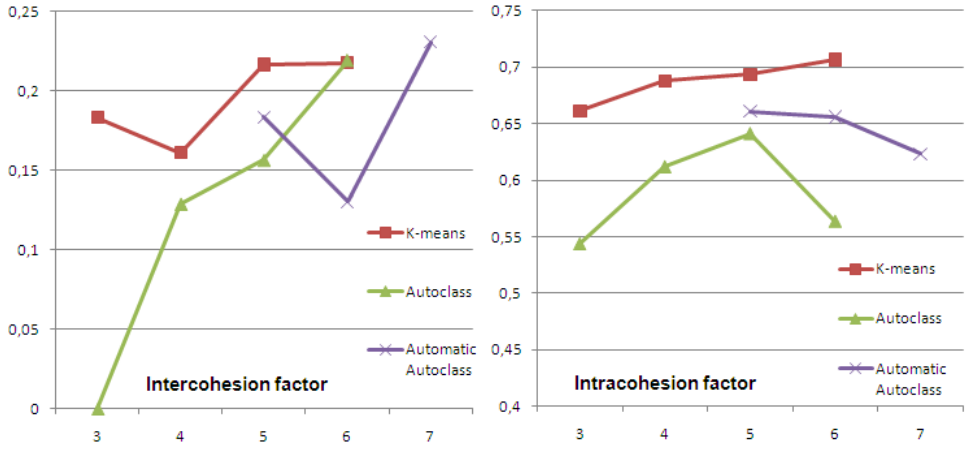


Figure 6.8: Intracohesion and Intercohesion evolution for K-means, Autoclass and automatic Autoclass.

In conclusion, the best number of clusters for the K-means approach applied to that security dataset is 4, as justified previously. SOM agrees with K-means. On the contrary, Autoclass would select 5 clusters as the best solution. However, after analyzing in detail the best solution of Autoclass, security analysts do not agree in some of the clusters. So the K-means option overtakes the Autoclass solution, providing better results.

Clustering approaches have also been applied to a more complete dataset of *Consensus*. This dataset included 90 devices tested in La Salle network, as detailed in Section 4.3. It included 46 computers from 4 student labs (14+12+12+8), 19 internal servers, 9 external servers, 10 staff computers and a computer cluster of 6 elements. The knowledge of this dataset was represented as a multidimensional vector with the list of all available ports and OS of the different devices of the security assessment.

K-means was executed with K values from 4 to 12 and 10 different random seeds. K-means results never got a solution with more than 9 clusters. Afterwards, execution results were evaluated with the validation indices to obtain the best solution with the best number of clusters. Figure 6.9

shows the evolution of the results of the all validation indices that are included in *Analia*: Dunn, DB, Silhouette, and the Cohesion factors for the executions with K-means.

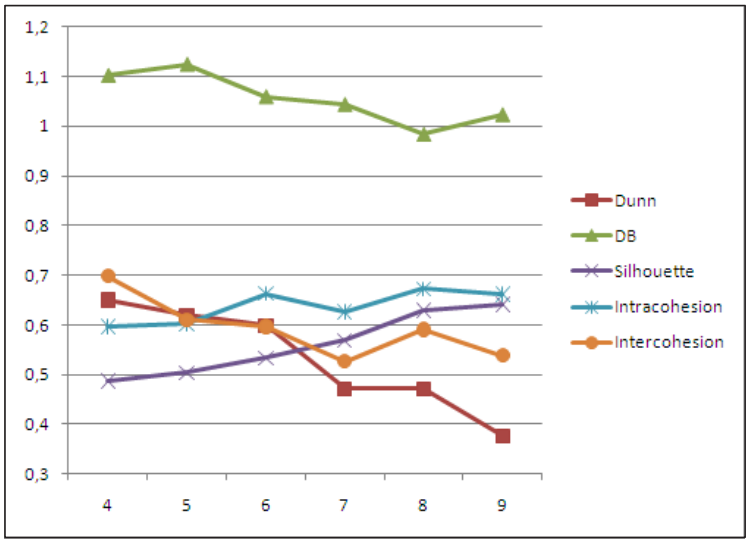


Figure 6.9: Validity index evolution for K-means.

Regarding Dunn index, higher values correspond to good cluster partitions. So the best solution would be 4 clusters. However this index is very sensitive to noise and tends to select few classes [56]. Therefore this conclusion will not be considered when it differs from the rest of the index results. Regarding DB index, low values correspond to good solutions. In this case, the best cluster would have 8 partitions. DB is considered more robust than Dunn validation, so 8 clusters are considered a better solution. Silhouette index has concluded with 9 clusters as the best solution, although the second best solution also has 8 clusters. Regarding Cohesion indices, Intracohesion factor agrees with DB solution in regarding 8 clusters as the best one. On the other hand, Intercohesion factor selects 7 clusters as the best option. It must be remarked that according to DB and Intracohesion, the best solution would have the most cohesive clusters in terms of operating systems, open services and common vulnerabilities. So this analysis comes to the conclusion that the best solution is to cluster the dataset into 8 groups, as it fulfills the initial requirements, although values close to 8 would be considered acceptable as well.

In addition, X-means was applied to the same dataset of 90 devices to check the clustering results results with the validation indices. X-means was executed varying the range of K values and using 10 random seeds. Figure 6.10 shows the evolution of validity indices for the different results of X-means. This figure reflects the best index value of the many solutions obtained for each number of clusters. As explained before, Dunn index results are not considered as they differ from the rest of the validity indices. All the other indices agree in the best solution for 7 clusters, although the Intercohesion factor also considers the solution of 9 clusters as a very good alternative. Again, the best solution has been evaluated by calculating the cohesion in terms of operating systems and open services by using DB and Silhouette indices, and also in terms of common vulnerabilities by using the Cohesion indices. So security analysts obtain a clustering solution that can be used to select the most critical cluster and start working with its devices to solve the harder security problems first.

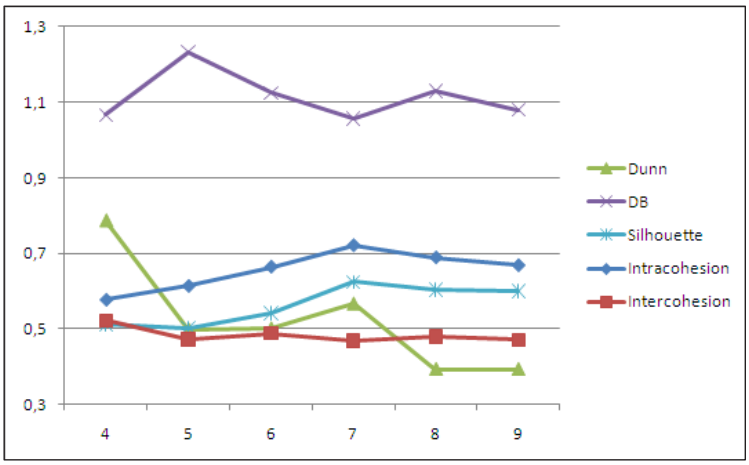


Figure 6.10: Validity index evolution for X-means.

6.6 Experimentation with weighted attributes

The experiments detailed in previous sections represented the *Consensus* knowledge with features related to open services and operating systems. This section details the analysis performed to introduce more features in the knowledge representation of tested devices.

Section 5.4 described the discussion about the selected pattern representations. When vulnerability data was analyzed to be included in the knowledge representation, a serious problem was encountered. Information related to the security level was represented by three different attributes: notes, warning and holes. However a complete representation of tested devices may contain many attributes in large networks with many devices. So including three attributes could go unnoticed when the number of the existing features was large enough. As a result, the information related to these attributes would not influence clustering results. This is why it was considered necessary to study the possibility of using weights in the attributes in order to change the low influence of security level features.

K-means, X-means and Autoclass do not allow the use of weights in the input parameters. In contrast, SOM can be configured to weight the input features by modifying the input configuration file. This is why this experimentation was carried out only with the SOM approach.

The knowledge representation was based on the pattern vector shown in Figure 5.16: a multidimensional vector with the list of all available ports, operating systems and security level features. The implemented weights were defined in 9 different input files. These weight files are the following:

- **View 0** (V_0): All the attributes have the same weight, except for the three security level attributes which have a weight = 0. This view is equivalent to the view used in the previous experimentations.

- **View 1** (V_1): All the attributes have the same weight, including the three security level attributes. So this configuration is just like adding new features to the whole input dataset.

- **View 2** (V_2): Attributes related to available ports have assigned a third of the total weight; attributes related to operating systems have a third of the total weight; the security level features have the other third assigned.

- **View 3** (V_3): Available ports, operating systems and security level features have a third of the total weight each. Regarding the security level features, Holes will be reweighted with

a half, whereas Warnings and Notes will be reweighted with a quarter from the initial third that had been assigned.

- **View 4 (V_4)**: Attributes related to available ports have assigned a quarter of the total weight; attributes related to operating systems have a quarter of the total weight; the security level features have the rest, that is the half of the total weight.

- **View 5 (V_5)**: Available ports and operating systems have a quarter of the total weight each. Regarding the security level features, Holes will be reweighted with a half, whereas Warnings and Notes will be reweighted with a quarter from the initial half that were assigned.

- **View 6 (V_6)**: Attributes related to available ports have assigned a 40% of the total weight; attributes related to operating systems have a 40% of the total weight; the security level features have the rest, the 20%.

- **View 7 (V_7)**: Available ports and operating systems features have a 40% of the total weight each. Regarding the security level features, Holes will be reweighted with a half, whereas Warnings and Notes will be reweighted with a quarter from the initial 20% that was assigned.

- **View 8 (V_8)**: Attributes related to available ports have assigned a half of the total weight; attributes related to operating systems have a half of the total weight; the security level features have no room in this configuration.

These different views have been evaluated with the experimental methodology proposed in [82], a general protocol for the study, from a statistical point of view, of the behavior of a set of problems. This methodology has concluded that the set formed by the views $\{V_0, V_1, V_2, V_3, V_6, V_7, V_8\}$ can not be considered significantly different in behavior in relation to V_6, which is the best configuration. However, it can be concluded that the views $\{V_4, V_5\}$ are significantly worse than V_6. These results are shown in Figure 6.11. The horizontal axis contains the range values for each configuration in ascendant direction. The Critical Distance (CD) has been calculated and equals 1,90. This CD is represented with a red line, showing that all views, except for views V_4 and V_5, are not significantly different from the best view, V_6, due to the fact that they are separated less than the critical distance.

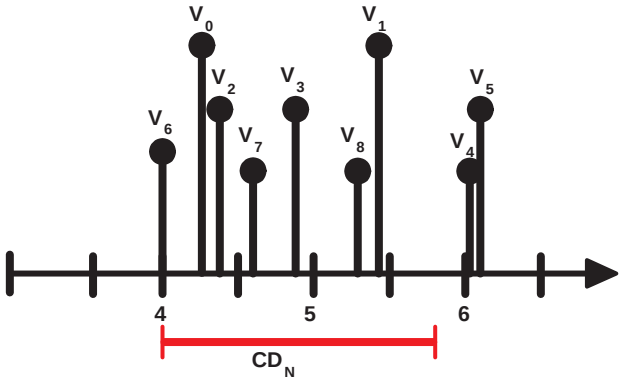


Figure 6.11: Representation of the analysis of the different weighted views.

6.7 Experimentation with multiobjective clustering

The experiments explained in this section analyze the performance of the multiobjective clustering approach compared to the other single-objective clustering approaches included in *Analia*. CAOS has been the selected algorithm [48]. When using clustering techniques like K-means, X-means, Autoclass or SOM, two steps have to be executed: first, one of the algorithms has to be selected for the executions and, afterwards, validation indices have to be applied to select the best execution. With the use of CAOS this process is shorten, due to the fact that validation indices are considered as initial goals of the clustering approach. Hence the resulting solution optimizes the selected indices in the clustering phase and not later on.

The input dataset has been extracted from *Consensus* database. This dataset was composed of 90 tested devices: 46 computers from 4 student labs (14+12+12+8), 19 internal servers, 9 external servers, 10 staff computers and a computer cluster for computational purposes of 6 components, as detailed in Section 4.3.

Overall deviation and connectivity are the two objectives to be optimized. The fulfillment of these objectives will result in a solution that maximizes the distance amongst clusters and minimizes the distances amongst elements in a same cluster.

The input parameters of CAOS [48] are the following:

- Maximum number of neighbors: 20
- Maximum external population: 1000
- Maximum internal population: 50
- Niche dimension: 0.2
- Maximum number of niches: 5
- Number of generations: 2000
- Crossover probability: 0.7
- Objectives: 2 - Overall deviation and connectivity
- Filter: Solutions that cluster all elements in a single cluster or solutions that cluster the population in more groups than the 10% of the population size are not considered. These solutions are not considered interesting as they would produce a single cluster or many clusters with few elements.

A solution for CAOS clustering run on *Consensus* dataset is shown in Figure 6.12. The best executions should find good tradeoffs between the two objectives and they are indicated by a circle centered around the solution. When analyzing the best clustering solutions, the number of clusters vary within 6 and 9. Some clusters are very clear and get repeated in all solutions. There is a cluster that always contains 14 PCs of lab1, 24 PCs of lab2 and 7 PCs of lab3, making a total of 45 devices. However this cluster should contain 46 devices, as lab3 was composed of 8 PCs. Then, it is very easy to discover that a device has been manipulated in that lab and the faked device is included in a cluster with a single element in all clustering solutions. Another big cluster is composed of all internal Linux servers and some external Linux servers. This cluster contains 27 devices that share the same operating system, although their open services are different. The rest of the devices are grouped in small clusters, depending on their operating system and the offered services. It is remarkable that 3 devices are separated always in 3 single clusters, showing their

Table 6.3: Summary of the *Cohesion* factors for K-means, X-means and CAOS clustering for different number of clusters.

Clusters	***K*-means**		***X*-means**		**CAOS Clustering**	
	Intrach.	Interch.	Intrach.	Interch	Intrach.	Interch.
6	0.662	0.597	0.559	0.546	0.858	0.451
7	0.626	0.526	0.628	0.538	0.879	0.419
8	0.672	0.591	0.621	0.531	0.895	0.380
9	0.662	0.538	0.563	0.517	0.866	0.371

dissimilarity in comparison with the rest of the elements in the dataset. They correspond to the fake device of a lab, to the wireless access control server which is a Linux device but with very specific peculiarities and, finally, to a Sun Solaris server.

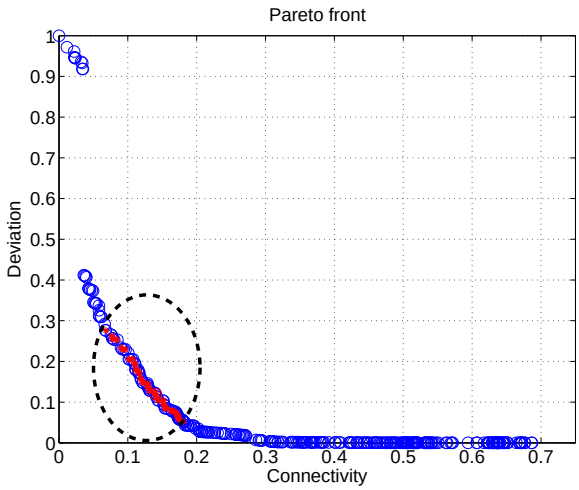


Figure 6.12: CAOS clustering output on *Consensus* with the deviation and the connectivity.

Cohesion factors have been calculated to evaluate the correctness of the different solutions and the alignment between these indices and CAOS objectives, overall deviation and connectivity. Results have shown that the best executions of CAOS also obtain the best values of *Cohesion factors*, achieving the best values of Intracohesion = 0.896 and Intercohesion = 0.380. The range of these indices is [0..1], preferring high values for *Intracohesion* and low values for *Intercohesion* indices [27].

Single-objective clustering algorithms have also been run on the same dataset to compare their solutions. K-means has run for a range of different numbers of clusters $K \in 3..10$ and different seeds. Considering *Cohesion* factors, the best solutions also obtain a number of clusters within 6 and 9. However, the calculated *Cohesion factors* are lower than CAOS results. The runs of X-means conclude in 7 for the best value of K. Again, the *Cohesion factors* are lower than CAOS clustering results. Both partitioning algorithms show clustering structures without one-element clusters, or at least, only one unique cluster with one element.

A summary of the obtained *Cohesion* factors of the different clustering approaches are shown in Table 6.3. The best values for all possible configurations of number of clusters are obtained with the CAOS approach. This is the approach that tries to optimize the two objectives more directly related to *Cohesion* factors. On the other hand, partition methods minimize only overall deviation and thus *Cohesion* factor results are not as good as the CAOS clustering approach [27].

Network security experts have also analyzed the results obtained after clustering the dataset.

Regarding CAOS clustering solution, the high number of clusters with a single element allows the location of outlier devices. But this approach returns a higher number of clusters for the same dataset, compared to the partitioning methods.

6.8 Experimentation with explanations of clustering results

The goal of the experimentation detailed in this section is to corroborate that clustering and explanations let security analysts obtain groups of devices with similar vulnerabilities and understand each cluster characterization. The experiments will show that explanations are also useful to explain the changes between a suspicious device and the rest of the elements, when an element is not clustered as it was expected.

The explanation process integrated in *Analia* is completely independent of the clustering technique used to partition the dataset. The process only considers the clustering output. All the clustering approaches included in *Analia* have been configured to extract their outputs in the same format and store them in *Consensus* database, providing a single entry point to the aforementioned process.

After clustering the dataset composed of 44 devices from La Salle network (see Section 6.4), explanations have been generated to justify clustering results. As an example, the explanations detailed in this section will refer to the output of SOM algorithm, although more experiments have been carried out with the output of the other clustering algorithms. The best executions have been selected using Cohesion factors as validity indexes. The explanation of the clustering solution for K=8 shown in Figure 6.6 will be detailed below.

The dataset included devices from 2 student labs, one with 14 devices and one with 7 devices. All of them should be Windows 2000 computers with exactly the same properties for the devices of the same lab. All should be grouped in the same cluster or divided into two clusters at most, one for each lab. However, the cluster solution for K=8 divided those elements into three clusters (see Figure 6.6). One cluster contained 14 devices of the same lab, as expected (devices 30-43). In contrast, the 7 devices of the second lab were placed in two different clusters, where one cluster contained six devices and another cluster contained a single element, the lab device 28. The cluster with 6 lab devices included also 2 internal Windows-2000 servers and a staff computer.

Figure 6.13 shows a different perspective of the resulting SOM executions for K=8. After obtaining clustering results from all executions, a statistic of how many times a device has been clustered with every other device for all executions has been calculated. The red circle in Figure 6.13 shows that device 28 has never been clustered with any other device for all the executions selected. On the other hand, devices 1, 2, 3, 4 and 26 have been clustered always together and device 27 has been clustered with them in more than the 80% of the executions. They are the devices of an student lab and they have been grouped together in the solution shown in Figure 6.6. In addition, the clustering distribution of devices shows that the computers of both labs have been grouped in the same cluster in the 20% of the executions, with the exception of device 28. Although analysts can easily detect an untrusting device, this information does not explain why this device is in a different cluster. However symbolic description of every cluster will give analysts this valuable information to quickly narrow the scope, isolate the device and apply the correction measures as soon as possible.

Table 6.4 shows a summary of the explanation of the clusters related to the student lab with 7 devices. Cluster 1 includes 6 of the 7 devices, whereas Cluster 2 is formed by a single element, the device 28. These explanations have been extracted from the output of the AU approach. In fact, the explanation of Cluster 2 shows all the features of device 28, as it is the only element in the cluster. In contrast, the explanation of Cluster 2 summarizes only the attributes common to all devices of that cluster. These descriptions show that the OS of cluster 1 has not been modified

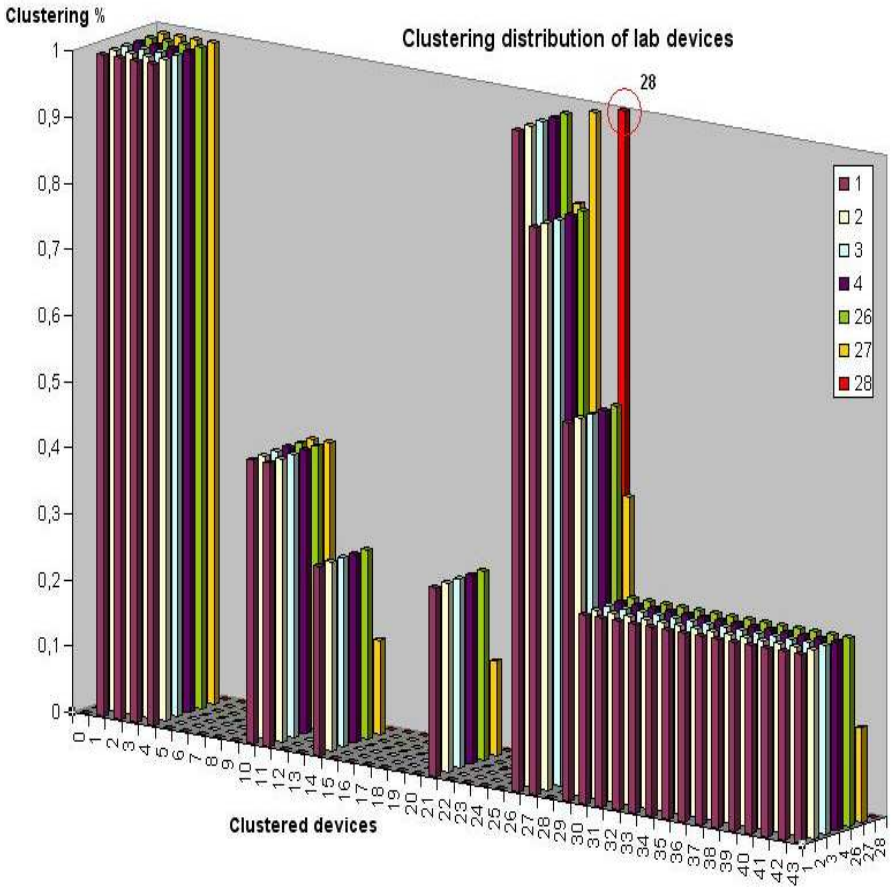


Figure 6.13: Clustering distribution of devices for SOM executions.

with respect to cluster 1. However cluster 2 contains a high number of filtered ports (value=2). A filtered port means that a firewall, filter or some other network obstacle is blocking the port and preventing the system from determining whether it is open. Another difference is found in port 135. Therefore, analysts can promptly act to solve this situation without having to analyze the vulnerabilities of all lab computers [23].

	Ports					**W2000**		**XP**		**W2003**	
Cluster	**135**	**139**	**445**	**781-807**	**904-930**	**WS SP4**	**S SP2**	**SP1**	**SP2**	**Standard**	**Enterprise**
C_1	—	1	1	—	—	67%	67%	67%	67%	70%	70%
C_2	1	1	1	2	2	67%	67%	67%	67%	70%	70%

Table 6.4: Explanation of two clusters obtained from AU algorithm.

A drawback of the explanations shown in Table 6.4 is that it is not possible to know whether the difference between both explanations is due to the fact that none of the elements of Cluster 1 have filtered ports or just some of them. This is why explanations have been created with the aid of DAU, the contribution of this thesis to improve explanations of clustering results (see Section 5.7.2). The results of applying DAU are shown in Table 6.5. The explanation of Cluster 1 shows that almost all the elements of that cluster coincided with the port 135 open (5 of 6 of the lab devices). So port 135 was not a decisive feature to separate device 28 from the rest of the lab devices. On the other hand the filtered ports detected in Cluster 2 are never shared with Cluster

1, providing a strong argument of separation of that device. DAU description also shows that Cluster 1 contains some devices that have altered services related to ports 7, 9, 13, 17, 19, 21, 25, 80 and 443. These changes have been not so relevant to separate those devices in another cluster, but give valuable information to analysts as all devices of Cluster 1 should contain exactly the same features with the same values. Consequently analysts can also detect that some devices in Cluster 1 have been modified as well [24].

	Ports								**W2000**		**XP**		**W2003**	
Cluster	**7,9, 13,17, 19,21, 25**	**80**	**135**	**139**	**443**	**445**	**781-807**	**904-930**	**WS SP4**	**S SP2**	**SP1**	**SP2**	**Stnd.**	**Entrp.**
C_1	1	1	1	1	1	1	—	—	67%	67%	67%	67%	70%	70%
	16%	33%	83%	100%	33%	100%	—	—	100%	100%	100%	100%	100%	100%
C_2	—	—	1	1	—	1	2	2	67%	67%	67%	67%	70%	70%
	—	—	100%	100%	—	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 6.5: Explanation of two clusters obtained from DAU algorithm.

Clustering has turned out to be a very usuful approach not only to identify similar devices, but also to find devices that unexpectedly appear separated and have new and different vulnerabilities. Explanations allow analysts to understand why some devices are in the same cluster or why they have been separated [23, 24]. This separation can be easily identified when a compact cluster with all the similar elements was expected. Also the relevant attributes of clusters are easily identified in order to detect the common vulnerabilities of all the devices included in a cluster. Therefore, when studying an element of a cluster, the obtained conclusions can be applied to the rest of its neighbors. Special efforts can be primarily focused on the most vulnerable clusters or the clusters where the most critical devices have been included.

6.9 Chapter summary

The experimentation performed with *Analia* to validate the different contributions has been detailed in this chapter. The experiments have been designed to verify all the features and enhancements included in *Analia* in order to improve the processes involved in the results analysis of security assessments.

The incorporation of clustering techniques in the analysis phase of security data has been experimented. Different knowledge representations have been used as input data for unsupervised learning approaches. Previous experiments have been run to gather data from security assessments. These experiments have been already explained in Chapter 4. Their execution has helped the system to compile enough data to be used as input parameters for clustering techniques. First tests have been executed with partition methods like K-means and X-means. A comparison of the results from both techniques and the arguments to justify the different clustering results are detailed. Subsequent experiments have been carried out with the Autoclass approach, both manually and automatically. Its results have also been compared with the K-means solution, achieving better results for K-means. SOM has been applied as a clustering technique due to its soft computing capabilities. Its results have been compared with K-means and X-means to demonstrate the usefulness of showing different points of view for the same dataset.

The need for a method to validate quantitatively clustering results forced us to experiment with different validity indices, including the contribution presented in this thesis: the Cohesion factors. These indices have been used not only to select the best clustering executions but also to calculate the best number of partitions when clustering outputs do not provide this feature.

The application of these indices has become a very useful method to compare clustering results of K-means, X-means, Autoclass and SOM.

A deeper study of the knowledge representations has been performed and the inclusion of new attributes in the representation pattern has been tried out. This study has been done irrespective of the clustering approach applied. Different weighting schemes have been tested. These new views of the same information have been evaluated and some conclusions can be extracted from this experiments. Some of the new knowledge representations have been discarded as they are not significantly better than the rest.

A new approach of clustering that integrates in a single step the clustering execution and the selection of the best execution has been included in *Analia*. This algorithm is CAOS and many tests have been executed. It seems obvious that *Analia* performance is improved when using a technique that agglutinates two phases in a single one, by using validation indices as objectives for clustering. However clustering results have to be corroborated as well, in order to check not only the performance improvement but also the quality of the clusters. The resulting clusters should group devices sharing OS, open ports and, consequently, sharing security vulnerabilities. Experimentation in this scene has focused on the confirmation of these results.

The final phase of the clustering process involves the use of explanations to describe the clustering output. Different experiments have been done to prove the usefulness of the contribution presented for *Analia* in this field: the DAU algorithm. This experimentation has been detailed and examples of explanations are given.

All the experimentation detailed in this section has focused on demonstrate that the use of unsupervised learning to deal with data from security assessments improves the daily work of security analysts. When using the unsupervised learning capabilities of *Analia*, security experts do not need to analyze the whole dataset to extract conclusions. Unsupervised learning will give them a value added service providing the crucial information from the whole dataset.

Part III

Conclusions and Future Work

Chapter 7

Conclusions and future work

The research of this thesis has concentrated its efforts on providing new contributions in the telematics field and, more specifically, in the field of network security. This thesis has proposed an approach between two areas that tend to follow separate ways: machine learning and network security. Moreover, both areas can profit from this collaboration. This chapter summarizes the work performed on the different lines of research and the contributions of this thesis. The future work is also outlined.

7.1 Framework of the thesis

This thesis has been integrated in the lines of research of the GRSI. The group focuses its research on Machine Learning, Case-Based Reasoning, Soft Computing and Evolutionary Computation to solve problems of classification and diagnosis, emphasizing the medical and telematics domains. These lines of research have colored the evolution of this thesis.

The main purpose of this thesis has been to provide a solution to improve the execution of security assessments within data networks and the subsequent analysis of the information gathered. With this in mind, existing security standards, procedures and methodologies have been surveyed. Complex and simple tools to help in security assessments have been also considered. This work has detected several limitations, so this thesis has contributed with an integrated framework for security assessments that follows a testing open-source methodology. This framework has been named *Consensus*. The study of the research lines connected to the GRSI more appropriate to our security framework has concluded in the use of some of the most relevant techniques to be included in *Analia*, the module in charge of analyzing testing results after a security assessment.

This chapter compiles the contributions defended during the thesis, as well as the papers and platforms that have been obtained as a result of the work on the different research lines.

7.2 Conclusions: contributions and work

The research framework of this thesis is in the field of data networks within the GRSI. Thus, the research contributions presented in this thesis answer various problems in telematics and in network security domains through the use and application of techniques of AI most appropriate for each scenario.

The security of data and communication networks is a huge domain with many lines of research. The work of this thesis focuses mainly on the study of the audits and security assessments that allow the detection of vulnerabilities in a network and in the elements that are part of it. In this specific environment, it has been detected a need for new mechanisms to help the automation of the processes associated with a security assessment following a recognized testing methodology. This process of automation is necessary, not only in the test phase but also in the analysis phase of

a security assessment. This main line of research falls into two sublines separated but coordinated: the first part provides proposals for automation of the various processes associated with the testing phase and it is developed in Chapter 3; on the other hand, the second part contributes to the automation of the analysis phase of a test, where the processing of data from a security test will be done through the application of AI and machine learning techniques. This second part is detailed in Chapter 5.

The contribution made in the first part for the automation of security testing processes has been materialized with the establishment of *Consensus*, a new platform to perform security tests. Furthermore, the contribution in the analysis phase of testing information has been reflected in the creation of a new system called *Analia*. The final contribution is a single integrated tool, since *Analia* is the result of incorporating AI techniques in the analysis module of *Consensus*. With these contributions, the time spent to gather information on the status of various network devices and the time spent to analyze such information is reduced. In addition, using a single tool to implement tests for the storage of data and for analysis of the results facilitates the management and the work associated with the security analysts and testers.

The contributions for each line of research are detailed in the next sections.

7.2.1 Automation of security assessments in data networks

Security is a feature of any system that indicates whether it is free from danger, harm or risk. Unfortunately the total security is an utopia, so there is always the likelihood of threats or dangers. Everyday new vulnerabilities and shortcomings that need to be solved arise. This is why control mechanisms are required to detect existing vulnerabilities periodically, in order to be able to handle them and apply the corrective measures.

Here are the different tasks performed in this thesis for this line of work:

- Analysis of the security regulations in the field of data networks to manage security information.
- Analysis of the security methodologies to carry out security tests and audits in communication networks.
- Analysis of the need for security tests to provide a proper security in a network. Study of the functionalities of the existing security tests.
- Analysis of the existing testing tools that facilitate the implementation of the regulation and security methodologies previously analyzed.
- Comparative analysis of methodologies and standards of network security.
- Analysis of the methodology OSSTMM.
- Analysis of the phases of a security test capable of being automated.
- Comparative analysis of security testing tools.
- Selection of the open-source security tools to be automated.
- Design of the architecture of a system capable of automating the process of testing data networks following the methodology OSSTMM. This system has been named *Consensus*.

  The design of *Consensus* is completely modular, scalable and adaptable to different network topologies. The design and description of its main features have been published in the following book:

- G. Corral, A. Zaballos, X. Cadenas, P. Herzog and I. Serra. "Consensus: Sistema Distribuido de Seguridad para el Testeo Automático de Vulnerabilidades". *Libro de Ponencias V Jornadas de Ingeniería Telemática*, 1: 351-358, 2005.

- Design of the Base System module. Definition of requirements, functionalities and security mechanisms.

- Design of the Management module. Definition of requirements and functionalities.

- Design of the Database module where the information gathered after a security assessment is stored. Definition of requirements and functionalities.

- Design of the Analysis Module. Definition of requirements and functionalities. Definition of the user interface.

- Description of the data set to be obtained from a device when it performs a security test.

- Design of the Internet Testing Module. Definition of requirements and functionalities. Selection of the security tools that integrate this module.

  This work was published in the journal of the *Universidad Politécnica de Cartagena* (UPCT) as an invited paper:

  - G. Corral, A. Zaballos, X. Cadenas and O. Prunera. "Automatización de un Sistema de Detección de Vulnerabilidades desde Internet". *III Teleco-Forum, UPCT 2005*, 3:9-12, 2005.

- Design of the Intranet Testing Module. Definition of requirements and functionalities. Selection of the security tools that integrate this module.

  The following publication contains the definition of this module and a description of its main features:

  - G. Corral, A. Zaballos, X. Cadenas and A. Grane. "A Distributed Vulnerability Detection System for an Intranet". *39th IEEE International Carnahan Conference on Security Technology (ICCST'05)*, 291-295, 2005.

- Design of the DMZ Testing Module. Definition of requirements and functionalities. Selection of the security tools that integrate this module.

- Design of the Wireless Testing Module. Definition of requirements and functionalities. Selection of the security tools that integrate this module. This module presents significant differences regarding the other modules, due to the fact that it is in charge of gathering information from wireless devices. The testing tools are specific for the wireless environment. The communication with the Management module has peculiarities to be considered. This module has been published in the following proceedings:

  - G. Corral, X. Cadenas, A. Zaballos and M. Cadenas. "A distributed vulnerability detection system for WLANs". *Proceedings of the 1st IEEE International Conference on Wireless Internet*, 86-93, 2005.

- Design of router and firewall vulnerability assessments. Definition of requirements and functionalities. Selection of the security tools that integrate this submodule.

- Design of the communications protocol among the different modules that form up *Consensus*. Definition of the control messages and analysis of its security. Definition of the messages for interchanging data among the testing modules and the Management module.
- Supervision and participation in the programming and implementation of the modules of *Consensus*.
- Experimentation with *Consensus* in real scenarios. Gathering of data and analysis of results.

In the research concerning the security environment, it has been highlighted the need of security testers, who ask for a system that integrates most of the existing tools, automates the testing process and stores information in a central repository, following a methodology like the OSSTMM. The solution to this requirement is the main contribution in this research line: the system *Consensus*.

Consensus allows the detection of security vulnerabilities in a network using a modular, independent, distributed and automated testing system. The different probes can test different types of networks, whether they are wired or wireless. The implementation of the platform *Consensus* has allowed the experience in real network environments, checking the minimization of time required to perform a test based on OSSTMM, when compared with the manual completion of the same type of test [30]. With *Consensus*, the tester installs simply the testing sensor and management elements. On the other hand, if security testers want to audit the network with simple open-source tools, it is necessary to install all of them individually, increasing the associated cost. Moreover, *Consensus* automates the call to these tools and stores the final results in the system automatically. Thus the results can be analyzed when security analysts consider it convenient, since data is never lost.

The main contributions have been accepted in different conferences and they have been published in their corresponding proceedings or in the journal of the UPCT [31]. Therefore, the whole framework *Consensus* has been presented [30] and the different testing modules of Intranet [29], Internet [31] and Wireless[25] have been published as well.

This first phase of automation has improved the processes associated with a security test of a data network and the storage of the information gathered. Nevertheless, the analysis of the results of each test must be done manually, analyzing the devices one by one. So, once settled the first stage, this research aims to improve the analysis phase of the data in a security assessment.

7.2.2 AI applied to the analysis of security in data networks

The second line of this research is a natural continuation of the previous line. The main objective is to improve the management of the information obtained in a security test, in order to help security analysts in the extraction of conclusions through a pre-processing of the information.

The automation of the steps associated with a security audit involves the improvement of the testing phase and the analysis phase. In the latter case, the objective is the application of machine learning techniques that allow extracting conclusions on data obtained from a test and, hence, it helps security analysts to manage the large volume of information.

The tasks undertaken in this research field are listed below:

- Analysis of the machine learning techniques more suitable for implementation in the data obtained from a security test.
- Analysis of the clustering strategies to structure the data stored in *Consensus*.
- Study and analysis of the K-means and X-means approaches.

- Definition and design of the knowledge representation used as input of the clustering algorithms.
- Selection of the most significant features that represent each device from all the data stored in *Consensus*.
- Experimentation with data files from *Consensus* and K-means and X-means.
- Analysis and validation of the results from K-means and X-means.

  The experimentation and analysis of results have been presented in the following book:

  - G. Corral, E. Golobardes, O. Andreu, I. Serra and E. Maluquer. "Application of Clustering Techniques in a Network Security Testing System". *Artificial Intelligence Research and Devolopment*, IOS Press, 131: 157-164, 2005.

- Design of the integration of the clustering algorithms in the analysis module of *Consensus*, named *Analia*.
- Participation and supervision of the implementation of the new analysis module for the integration of K-means and X-means.
- Analysis of Autoclass as an alternative of K-means or X-means. Experimentation and drawing conclusions.
- Study and analysis of SOM to manage data from security assessments. Experimentation with SOM and data stored in *Consensus*.
- Analysis and validation of the results obtained after applying SOM in data files that include the significant attributes of the devices tested previously with *Consensus*.
- Comparison of the results obtained after the application of K-means, X-means and SOM.

  The same data set, retrieved from the *Consensus* database, has been clustered using different unsupervised learning techniques: K-means, X-means and SOM. We have conducted a comparison of different solutions and the results have been presented in the following publications. The first paper presents a qualitative comparison among the three options [47], while the second publication presents a quantitative comparison of the results [26]. The publications are listed below:

  - A. Fornells, E. Golobardes, D. Vernet and G. Corral. "Unsupervised Case Memory Organization: Analysing Computational Time and Soft Computing Capabilities". *Lecture Notes in Computer Science (LNCS): Advances in Case-Based Reasoning*, Springer, 4106:241-255, 2006.
  - G. Corral, A. Fornells, E. Golobardes and J. Abella. "Cohesion factors: improving the clustering capabilities of Consensus". *Lecture Notes in Computer Science (LNCS): Intelligent Data Engineering and Automated Learning - IDEAL 2006*, Springer, 4224:488-495, 2006.

- Study and analysis of a Multiobjective Clustering approach, CAOS, to manage data from security assessments. Experimentation with CAOS and data stored in *Consensus*.
- Comparison of the results obtained after the application of CAOS and the other clustering approaches detailed before [27]. A publication on this topic is listed below:

  - G. Corral, A. Garcia-Piquer, A. Orriols-Puig, A. Fornells and E. Golobardes. "Multi objective Evolutionary Clustering Approach to Security Vulnerability Assesments". In *4th International Conference on Hybrid Artificial Intelligence Systems*, 2009, Lecture Notes in Computer Science, Springer, 597-604, 2009.

- Analysis of estrategies for cluster validation.
- Design of new indices for cluster validation in order to assess quantitatively, from a telematics viewpoint, the performance of the results obtained with different clustering techniques. These indexes are named *Cohesion factors*.
- Implementation of the *Cohesion factors* and integration of these indices in *Analia*.
- Application of the *Cohesion factors* to the clustering results obtained previously when using K-means, X-means, Autoclass, SOM and CAOS.
- Analysis of the *Cohesion factors* and comparison of the solutions obtained of the clustering techniques mentioned above.

  The calculation of the *Cohesion factors* on the results obtained with different clustering techniques provides a quantitative tool to assess the best distribution. It is also useful to check that the clusters obtained, where devices with similar ports and OS are grouped, also contain devices with similar vulnerabilities. The *Cohesion factors* and a comparison of different solutions have been presented in the following publications:

  - G. Corral, A. Fornells, E. Golobardes and J. Abella. "Cohesion factors: improving the clustering capabilities of Consensus". *Lecture Notes in Computer Science (LNCS): Intelligent Data Engineering and Automated Learning - IDEAL 2006*, Springer, 4224:488-495, 2006.
  - G. Corral, A. Garcia-Piquer, A. Orriols-Puig, A. Fornells and E. Golobardes. "Multi objective Evolutionary Clustering Approach to Security Vulnerability Assesments". *In 4th International Conference on Hybrid Artificial Intelligence Systems*, Lecture Notes in Computer Science, Springer, 597-604, 2009.

- Proposal for automatic calculation of the best number of clusters to be obtained for each data set based on the *Cohesion factors*.

  Techniques like K-means and SOM need to know in advance the number of clusters in which you want to split the dataset into. This information is not known at the beginning of execution by being in an unsupervised environment. The *Cohesion factors* may help to obtain the best clustering solution and consequently provide better value for groups, given different options. This proposal was included in the following publication:

  - G. Corral, A. Fornells, E. Golobardes and J. Abella. "Cohesion factors: improving the clustering capabilities of Consensus". *Lecture Notes in Computer Science (LNCS): Intelligent Data Engineering and Automated Learning - IDEAL 2006*, Springer, 4224:488-495, 2006.

- Analysis of strategies for explanations of clustering results.
- Adaptation of AU to *Analia domain*.
- Design of a new algorithm based on AU to explain clustering results after applying a clustering algorithm, named DAU.

- Implementation of AU and DAU and integration of these algorithms in *Analia*.

  This work has been published as detailed below:

  - G. Corral, E. Armengol, A. Fornells, and E. Golobardes. "Explanations of Unsupervised Learning Clustering applied to Data Security Analysis". *Neurocomputing*, Elsevier, 72:2754-2762, 2009.
  - G. Corral, E. Armengol, A. Fornells and E. Golobardes. "Data Security Analysis using Unsupervised Learning and Symbolic Descriptions". In *Innovations in Hybrid Intelligent Systems*, Vol. 44, pages 112-119, 2007, Advances in Soft-Computing.

This line of work has produced different contributions. A first contribution is the use of unsupervised learning for processing information obtained from a security test. It was necessary to define a valid representation of knowledge to be used as input of clustering algorithms. A thorough survey in the literature has shown that these techniques had not been used previously to process such information. It has been shown to be useful in the analysis of the results of a security assessment, due to the fact that it makes it possible to extract some conclusions about the entire volume of information quickly. The grouping of devices with similar vulnerabilities allows security analysts to prioritize groups of machines on which to work with. Moreover, if analysts know the features of the original device, they can get to find elements that have experienced intrusions or changes to appear together with other devices that, theoretically, are not similar. Even these devices could be altered enough to form a new cluster, fact that would alert analysts easily.

The second contribution of this research is *Analia*, the new system that incorporates techniques of unsupervised learning in the analysis module of *Consensus*. The integration of the stages of test and analysis in one tool facilitates the management of experts in security and allows further improvements in a single platform.

Another important aspect is the fact that the results of the clustering techniques are really good for security analysts. Experts could manually validate the different groups of devices, but this is not a viable option if, in fact, the ultimate goal is to automate the process so as to relieve security experts of tedious tasks. Therefore, it is important to have an objective and quantitative measure that allows analysts to validate that the resulting clusters are devices with similar vulnerabilities, although they have been classified according to criteria of services and operating systems. The third contribution of this research includes the creation of new indices of validation, the *Cohesion factors*. These factors measure the compactness of each cluster and validate the separation between different clusters [26]. The *Cohesion factors* are useful to validate the results of the clustering solution obtained. At the same time, these indices can be used to calculate the optimal number of clusters, so as to improve the performance of algorithms like K-means and SOM. This automatic calculation of the best number of clusters is the fourth contribution in this research.

The last contribution is related to the explanation of clustering results [24, 27]. This line of research has been focused on the use of clustering and generalizations for knowledge discovery. These generalizations are symbolic descriptions about a group of elements and they describe why a subset of cases has been grouped together. These generalizations are based on the antiunification concept to characterize each cluster. Broadly speaking, they contain a description with the attributes shared by both objects whose value is the most specific. As for generalizations, we have named them explanations. These explanations allow describing a cluster with the same representation language used to describe the elements; therefore security analysts can more easily understand results. Consequently, analysts obtain a solution where all tested devices have been grouped in different clusters regarding their vulnerabilities and know the reason of these clustering results. The application of clustering and explanations help analysts to obtain groups of devices

with similar vulnerabilities, to understand each cluster characterization and to detect unauthorized changes in network devices.

7.3 Summary of publications

This section lists the different papers that have been published in relation to the main goal of this thesis: to contribute with a security framework to improve the security assessment of data networks.

- G. Corral, A. Garcia-Piquer, A. Orriols-Puig, A. Fornells and E. Golobardes. "Multiobjective Evolutionary Clustering Approach to Security Vulnerability Assesments". In *4th International Conference on Hybrid Artificial Intelligence Systems*, Lecture Notes in Computer Science, Springer, 597-604, 2009.

- G. Corral, E. Armengol, A. Fornells, and E. Golobardes. "Explanations of Unsupervised Learning Clustering applied to Data Security Analysis". Neurocomputing, Elsevier, 72:2754-2762, 2009.

- G. Corral, E. Armengol, A. Fornells and E. Golobardes. "Data Security Analysis using Unsupervised Learning and Symbolic Descriptions". In *Innovations in Hybrid Intelligent Systems*, Vol. 44, pages 112-119, 2007, Advances in Soft-Computing.

- G. Corral, A. Fornells, E. Golobardes and J. Abella. "Cohesion factors: improving the clustering capabilities of Consensus". In *7th International Conference on Intelligent Data Engineering and Automated Learning*, Vol. 4224, pages 488-495, 2006, Lecture Notes in Computer Science, Springer-Verlag.

- A. Fornells, E. Golobardes, D. Vernet and G. Corral. "Unsupervised Case Memory Organization: Analysing Computational Time and Soft Computing Capabilities". In *8th European Conference on Case-Based Reasoning*, Vol. 4106, pages 241-255, 2006, Lecture Notes in Computer Science, Springer-Verlag.

- G. Corral, X. Cadenas, A. Zaballos and M. Cadenas, "A Distributed Vulnerability Detection System for WLANs". In *First International Conference on Wireless Internet - WICON 2005*, pages 86-93, 2005, IEEE Computer Society.

- G. Corral, A. Zaballos, X. Cadenas and A. Grane. "A Distributed Vulnerability Detection System for an Intranet". *39th IEEE International Carnahan Conference on Security Technology (ICCST'05)*, 291-295, 2005.

- G. Corral, A. Zaballos, X. Cadenas and O. Prunera. "Automatización de un Sistema de Detección de Vulnerabilidades desde Internet". *III Teleco-Forum, UPCT 2005*, 3:9-12, 2005.

- G. Corral, A. Zaballos, X. Cadenas, P. Herzog and I. Serra. "Consensus: Sistema Distribuido de Seguridad para el Testeo Automático de Vulnerabilidades". *Libro de Ponencias V Jornadas de Ingeniería Telemática*, 1: 351-358, 2005.

- G. Corral, E. Golobardes, O. Andreu, I. Serra, E. Maluquer and A. Martínez. "Application of Clustering Techniques in a Network Security Testing System". In *Artificial Intelligence Research and Devolopment*,131:157-164, 2005, IOS Press.

7.4 Future work

In the previous section, the contributions made in the various lines of research have been described. Security professionals have a tough challenge coping with the vast amount of alerts and logs that are produced in a network. They often come from multiple sensors, spanning multiple complex subsystems. This complexity implies that such systems require constant monitoring and maintenance. Human capacities are not sufficient and a system like the platform proposed in this thesis, *Consensus*, tries to address these issues.

Clustering techniques have been selected for the analysis of information security, like K-means, X-means, Autoclass, and CAOS, together with self-organizing maps. In different fields, the usefulness and the improvements have been demonstrated through the application of these methods. This approach between telematics and machine learning fields provide alternative solutions to existing problems.

All significant results have been reflected in the completion of publications, both in the fields of telematics and artificial intelligence. So, the initial objectives have been achieved and it has been proved that the application of machine learning in data network problems provide benefits and improvements.

Regarding future work, there is still a roadmap to keep on working and approaching even more the connection between the telematics and the machine learning worlds. Here are the main current and future works.

Information gathered of each device after a security test is very extensive. The contributions presented in this thesis operate with three main types of attributes to represent each device: open services, operating systems and vulnerability information. All these attributes are numerical. However, many data regarding vulnerabilities is stored in strings. These strings cannot be used as input features due to the fact that the clustering algorithms included in *Analia* do not work with string attributes. Therefore, a future line of work is the analysis and design of other knowledge representations that include these kind of attributes to be parameterized.

The improvement of the multiobjective clustering approach based on genetic algorithms, CAOS, is one of the most immediate future works. The experimentation with the typical objectives (overall deviation and connectivity) has shown positive results. Therefore, new objectives will be included as input, in order to corroborate the goodness of its results when using validity indices like the *Cohesion factors*. Related to this point, research will continue regarding cluster validation. It is imperative to have tools for validating the results which confirm that the resulting clusters are effective and useful for security analysts. We will continue researching in this direction, with the study and analysis of statistical techniques for validation of results that can be adapted to *Analia*. In fact, this research line is aligned with the objectives of the project submitted for the next national call of research projects. This project is named NEXT-CBR, which is the evolution of the actual research project MID-CBR. In this project, topics in the case retrieval stage using soft-CBR applied to the telematics domain are completely aligned with this future work. However the official notification of acceptance has not arrived yet.

The discovery of common patterns between the data gathered from the devices after a security test may allow to separate the different examples in classes. In this case, it will be possible to design and create a CBR system that stores that information to solve a new situation, which is the detection of vulnerabilities in a new device. So, this future work will require a comprehensive study of the various phases of a CBR system in order to adapt them to the application environment.

Part IV

Bibliography

Bibliography

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundations issues, methodological variations, and system approaches. *IA Communications*, 7:39–59, 1994.

[2] R. Agrawal, J. Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[3] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. *In Proceedings of 9th ACM Conference on Computer and Communications Security*, 2002.

[4] K. Anchor, J. Zydallis, and G. Gunsch. Extending the computer defense immune system: Network intrusion detection with a multiobjective evolutionary programming approach. In *1st Conf. on Artificial Immune Systems*, pages 12–21, 2002.

[5] J.P. Anderson. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Co., Fort Washington*, 1980.

[6] M. Ankerst, M.M. Breunig, H. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 49–60. ACM Press, 1999.

[7] Attribute relation file format. http://www.cs.waikato.ac.nz/ ml/weka/arff.html.

[8] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning*, 41(3):259–294, 2000.

[9] E. Armengol and E. Plaza. Symbolic explanation of similarities in cbr. *Computing and Informatics*, 25:1001–1019, 2006.

[10] S. Bandyopadhyay and U. Maulik. Nonparametric genetic clustering: Comparison of validity indices. In *IEEE Transactions on systems, man, and cybernetics*, volume 31, pages 120–125, 2001.

[11] M.J. Berry and G. Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons, Inc., 1997.

[12] K. Nasihin bin Baharin, N. Md Din, Md Zaini Jamaludin, and Nooritawati Md Tahi. Third party security audit procedure for network environment. *Proceedings of the 4th Conference on Telecommunication Technology. NCTT 2003*, pages 26–30, 2003.

[13] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[14] M. Bélanger and J. Martel. An automated explanation approach for a decision support system based on mcda. In Thomas Roth-Berghofer and Stefan Schulz, editors, *ExaCt*, volume FS-05-04 of *AAAI Technical Report*, pages 21–34. AAAI Press, 2005.

[15] E. Bloedorn, L. Talbot, and D. DeBarr. *Data Mining Applied to Intrusion Detection: MITRE Experiences.* Marcus A. Maloof, ed., Springer Verlag, New York, 2005.

[16] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vision Graph. Image Process.*, 37(1):54–115, 1987.

[17] J. Cassens. Knowing what to explain and when. In P. Gervás and K. Gupta eds., editors, *Proceedings of ECCBR Workshop*, pages 97–104, 2004.

[18] Cert coordination center statistics, 2008. www.cert.org/stats/cert_stats.html.

[19] R. Chandramouli, T. Grance, R. Kuhn, and S. Landau. Common vulnerability scoring system. In *IEEE Security and Privacy*, pages 85–89. IEEE Computer Society, 2006.

[20] D. Chapman and E. Zwicky. O'Reilly, 2000.

[21] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180, 1996.

[22] S. Chen, J. Xu, Z. Kalbarczyk, and R.K. Iyer. Security Vulnerabilities: From Analysis to Detection and Masking Techniques. *Proceedings of the IEEE*, 94:407–418, 2006.

[23] G. Corral, E. Armengol, A. Fornells, and E. Golobardes. Data security analysis using unsupervised learning and explanations. In *Innovations in Hybrid Intelligent Systems*, volume 44 of *Advances in Soft Computing*, pages 112–119. Springer, 2008.

[24] G. Corral, E. Armengol, A. Fornells, and E. Golobardes. Explanations of unsupervised learning clustering applied to data security analysis. *Neurocomputing*, 72(13-15):2754–2762, August 2009.

[25] G. Corral, X. Cadenas, A. Zaballos, and M. Cadenas. A distributed vulnerability detection system for WLANs. In *Proceedings of the First International Conference on Wireless Internet - WICON 2005*, pages 86–93, 2005.

[26] G. Corral, A. Fornells, E. Golobardes, and J. Abella. Cohesion factors: improving the clustering capabilities of consensus. In *Intelligent Data Engineering and Automated Learning - IDEAL 2006, in Lecture Notes in Computer Science*, volume 4224, pages 488–495. Springer, 2006.

[27] G. Corral, A. Garcia-Piquer, A. Orriols-Puig, A. Fornells, and E. Golobardes. Multiobjective evolutionary clustering approach to security vulnerability assessments. In *Hybrid Artificial Intelligence Systems*, Lecture Notes in Artificial Intelligence, pages 597–604. Springer, 2009.

[28] G. Corral, E. Golobardes, O. Andreu, I. Serra, E. Maluquer, and A. Martínez. Application of clustering techniques in a network security testing system. *Artificial Intelligence Research and Devolopment*, 131:157–164, 2005.

[29] G. Corral, A. Zaballos, X. Cadenas, and A. Grane. A distributed vulnerability detection system for an intranet. In *Proceedings of the 39th IEEE International Carnahan Conference on Security Technology (ICCST'05)*, pages 291–295, 2005.

[30] G. Corral, A. Zaballos, X. Cadenas, P. Herzog, and I. Serra. Consensus: Sistema distribuido de seguridad para el testeo automático de vulnerabilidades. *V Jornadas de Ingeniería Telemática Jitel'05*, pages 351–358, 2005.

[31] G. Corral, A. Zaballos, X. Cadenas, and O. Prunera. Automatización de un sistema de detección de vulnerabilidades desde internet. *III Teleco-Forum, UPCT*, pages 9–12, 2005.

[32] G. Corral, A. Zaballos, J. Camps, and J.M. Garrell. Prediction and control of short-term congestion in ATM networks using Artificial Intelligence techniques. In *Lecture Notes In Computer Science: Proceedings of the First International Conference on Networking-Part 2*, volume 2094, pages 648–657, London, UK, 2001. Springer-Verlag.

[33] D.L. Davies and D.W. Bouldin. A cluster separation measure. In *IEEE Transactions on Pattern Analysis and Machine Learning*, volume 4, pages 224–227, 1979.

[34] J. Dawkins and J. Dale. A systematic approach to multi-stage network attack analysis. *Proceedings of the 2nd. IEEE International Information Assurance Workshop(IWIA'04)*, 2004.

[35] G. DeJong. Explanation-based learning. In Allen B. Tucker, editor, *The Computer Science and Engineering Handbook*. CRC Press, 2004.

[36] L. DeLooze. Classification of computer attacks using a self-organizing map. In *Proc. of the 2004 IEEE Workshop on Information Assurance*, pages 365–369, 2004.

[37] M.O. Depren, M. Topallar, E. Anarim, and K. Ciliz. Network-based anomaly intrusion detection system using soms. In *Proc. of the IEEE 12th Signal Processing and Communications Applications Conference*, pages 76–79, 2004.

[38] D. Doyle, P. Cunningham, D.G. Bridge, and Y. Rahman. Explanation oriented retrieval. In P. Funk and P.A. González-Calero, editors, *ECCBR*, volume 3155 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 2004.

[39] Y. Du and D. Hoffman. Pbit: A pattern-based testing framework for iptables. In *CNSR '04: Proceedings of the Second Annual Conference on Communication Networks and Services Research*, pages 107–112, Washington, DC, USA, 2004. IEEE Computer Society.

[40] J.C. Dunn. Well separated clusters and optimal fuzzy partitions. In *Journal of Cybernetics*, volume 4, pages 95–104, 1974.

[41] A. El-Atawy, K. Ibrahim, H. Hamed, and E. Al-Shaer. Policy segmentation for intelligent firewall testing. *1st IEEE ICNP Workshop on Secure Network Protocols*, pages 67–72, 2005.

[42] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data, 2002.

[43] M. Ester, H.P. Kriegel, and X. Xu. A database interface for clustering in large spatial databases. In *Knowledge Discovery and Data Mining*, pages 94–99, 1995.

[44] Estudio sobre el estado actual de la seguridad de los sistemas de información en las empresas entrevistadas, de acuerdo con la norma ISO/IEC 17799:2000. http://alerta-antivirus.red.es/docs/seguridad_empresas.pdf.

[45] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. *Lecture Notes in Computer Science*, 2259, 2001.

[46] A. Fornells. *Marc integrador d'estratègies Soft Computing pel desenvolupament de sistemes basats en el Raonament Basat en Casos*. PhD thesis, Enginyeria i Arquitectura La Salle (Universitat Ramon Llull), 2007.

[47] A. Fornells, E. Golobardes, D. Vernet, and G. Corral. Unsupervised case memory organization: Analysing computational time and soft computing capabilities. In Springer, editor, *Lecture Notes in Computer Science: Advances in Case-Based Reasoning*, volume 4106/2006, pages 241–255, 2006.

[48] A. Garcia-Piquer. *Diploma d'Estudis Avançats*. Enginyeria i Arquitectura La Salle (Universitat Ramon Llull), 2009.

[49] D. Geer and J. Harthorne. Penetration testing: a duet. *Proceedings of the 18th Computer Security Applications Conference*, pages 185–195, 2002.

[50] David P. Gilliam, John C. Kelly, John D. Powell, and Matt Bishop. Development of a software security assessment instrument to reduce software security risk. *wetice*, page 144, 2001.

[51] E. Golobardes and E. Bernado. Apunts del Curs de Doctorat Aprenentatge artificial i representació del coneixement, Universitat Ramon Llull, 2005. http://www.salleurl.edu/∼ elisabet/Assignatures/AA/temari_ml.html.

[52] M. Gu, A. Aamodt, and X. Tong. Component retrieval using conversational case-based reasoning. pages 259–271, 2005.

[53] S. Gunter and H. Burke. Validation indices for graph clustering. In *Proc. 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 229–234, 2001.

[54] F. Guo, Y. Yu, and T. Chiueh. Automated and Safe Vulnerability Assessment. *21st Annual Computer Security Applications Conference ACSAC'05*, 2005.

[55] M. Gupta, J. Rees, A. Chaturvedi, and J. Chi. Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach. *Decision Support Systems*, 41(3):592–603, March 2006.

[56] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. In *Journal of Intelligent Information Systems*, volume 17, pages 107–145, 2004.

[57] J. Han and M. Kamber. Data mining: Concepts and techniques, 2000.

[58] J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. In *Technical Report TR-COMPSYSBIO-2004-02*. UMIST, 2004.

[59] J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76, Feb. 2007.

[60] J. Hartigan and M. Wong. A k-means clustering algorithm. In *Applied Statistics*, pages 28:100–108, 1979.

[61] J.A. Hartigan. *Clustering Algorithms*. John Wiley and Sons, New York, 1975.

[62] A. Hinneburg and D.A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[63] L. Hubert and J. Schultz. Quadratic assignment as a general data-analysis strategy. In *British Journal of Mathematical and Statistical Psychologie*, volume 29, pages 190–241, 1976.

[64] Intrusion Detection Exchange Format (IDWG). 59th IETF Meeting, 2003.

[65] Top 100 Network Security Tools - by Insecure.org, 2006. http://sectools.org.

[66] Registre de certificacions iso/iec 17799 i iso/iec 27001. http://www.iso27001certificates.com.

[67] P. Herzog (ISECOM). Open source security testing methodology manual 2.1 (OSSTMM), 2004. http://www.isecom.org/osstmm/.

[68] K. Jiwnani and M. Zelkowitz. Susceptibility matrix: A new aid to software auditing. *IEEE Security and Privacy*, 2(2):16–21, 2004.

[69] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, 6:443–471, 2003.

[70] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen. Analysis of vulnerabilities in internet firewalls. *Computers and Security*, 22:214–232, 2003.

[71] J. Kim, Y.K. Malaiya, and I. Ray. Vulnerability discovery in multi-version software systems. In *HASE '07: Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium*, pages 141–148, Washington, DC, USA, 2007. IEEE Computer Society.

[72] T. Kohonen. *Self-Organizing Maps*. Springer, 3rd. Edition, 2000.

[73] I. Kotenko. Active vulnerability assessment of computer networks by simulation of complex remote attacks. *International Conference on Computer Networks and Mobile Computing*, pages 40–47, 2003.

[74] U. Larson, E. Lundin-Barse, and E. Johnsson. METAL - A Tool for Extracting Attack Manifestations. *LNCS: Intrusion and Malware Detection and Vulnerability Assessment*, pages 85–102, 2002.

[75] A. Lee, G. Koenig, X. Meng, and W. Yurcik. Searching for open windows and unlocked doors: Port scanning in large-scale commodity clusters. *IEEE International Symposium on Cluster Computing and Grid*, 1:146–151, 2005.

[76] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proc. 28th Australasian CS Conf.*, volume 38 of *CRPITV*, 2005.

[77] Y. Lim, T.R. Schmoyer, J. Levine, and H. Owen. Wireless intrusion detection and response. In *Proceedings of the Information Assurance Workshop*, pages 68–75, 2003.

[78] E. C. Lo and M. Marchand. Security audit: a case study [information systems]. In *Canadian conference on Electrical and Computer Engineering*, pages 193–196. IEEE Computer Society, 2004.

[79] M. R. Lyu and L. K. Lau. Firewall security: Policies, testing and performance evaluation. In *COMPSAC '00: 24th International Computer Software and Applications Conference*, pages 116–121, Washington, DC, USA, 2000. IEEE Computer Society.

[80] D. Marchette. A statistical method for profiling network traffic. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 119–128, Santa Clara, CA, 1999.

[81] F. Martin. *Case-Based Sequence Analysis in Dynamic, Imprecise, and Adversarial Domains*. PhD thesis, Universitat Politècnica de Catalunya, 2004.

[82] J.M. Martorell. *Definició d'una metodologia experimental per a l'estudi de resultats en sistemes d'aprenentatge artificial*. PhD thesis, Enginyeria i Arquitectura La Salle (Universitat Ramon Llull), 2007.

[83] M.Boldt and B. Carlsson. Analysing Privacy-Invasive Software Using Computer Forensic methods. *DIMVA06*, 2006.

[84] L. Mcginty and B. Smyth. *On the Role of Diversity in Conversational Recommender Systems*. 2003.

[85] D. McSherry. Explanation in recommender systems. In *Artificial Intelligence Review*, volume 24, pages 179–197, 2005.

[86] R.S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*.

[87] MITRE. CVE: Common vulnerabilities and esposures, 2008. http://www.cve.mitre.org/.

[88] MITRE. OSVDB: The open source vulnerability database, 2008. http://osvdb.org/.

[89] A.M. Baith Mohamed. An effective modified security auditing tool (sat). *Proceedings of the 23rd International Conference on Information Technology Interfaces. ITI 2001*, pages 37–41, 2001.

[90] B. Morin, L. Me, H. DeBar, and M. Ducasse. M2d2: A formal data model for ids alert correlation. In *Recent Advances in Intrusion Detection, LNCS*, volume 2516, pages 115–137, 2002.

[91] K. Nauta and F. Lieble. Offline network intrusion detection: Looking for footprints. *SAS White Paper*, 2000.

[92] M. Nicolau, J. Abella, G. Corral, and F. Garcia. Algoritmo distribuido de reconfiguración de redes virtuales ATM basadas en VPs. *II Jornadas de Ingeniería Telemática Jitel'99*, pages 381–388, 1999.

[93] C. Nugent and P. Cunningham. A case-based explanation system for black-box systems. *Artificial Intelligence Review*, 24(2):163–178, 2005.

[94] OCTAVE (operationally critical threat, asset, and vulnerability evaluation). http://www.cert.org/octave.

[95] X. Ou, S. Govindavajhala, and A.W. Appel. MulVAL: A Logic-based Network Security Analyzer. *Proceedings of the 14th USENIX Security Symposium*, 2005.

[96] D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference of Machine Learning (ICML)*, pages 727–734. Morgan Kaufmann Publishers Inc., 2000.

[97] T.R. Peltier, J. Peltier, and J. Blackley. *Managing a Network Vulnerability Assessment*. Auerbach Publishers Inc., 2003.

[98] C. Phillips and L Swiler. A graph-based system for network-vulnerability analysis. *In Proceedings of the Workshop on New Security Paradigms*, pages 71–79, 1998.

[99] J.D. Pierce, A.G. Jones, and M.J. Warren. Penetration testing professional ethics: A conceptual model and taxonomy. *Australasian Journal of Information Systems*, 13(2):193–200, 2006.

[100] B.G. Buchanan R. Davis and E.H. Shortliffe. Production systems as a representation for a knowledge-based consultation program. *Artificial Intelligence*, 8:15–45, 1977.

[101] M. Ramadas, S. Ostermann, and B. C. Tjaden. Detecting anomalous network traffic with self-organizing maps. In *RAID'03:Proc. 6th Symposium on Recent Advances in Intrusion Detection*, volume 2820, pages 36–54, 2003.

[102] R.W. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. *In IEEE Symposium on Security and Privacy*, pages 156–165, 2000.

[103] P.J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. In *Journal of Computational Applications in Math*, volume 20, pages 53–65, 1987.

[104] A. Sharma, J. R. Martin, N. Anand, M. Cukier, and W.H. Sanders. Ferret: A host vulnerability checking tool. In *Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04)*, pages 389–394, Washington, DC, USA, 2004. IEEE Computer Society.

[105] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of 24th International Conference Very Large Data Bases, VLDB*, pages 428–439, 24–27 1998.

[106] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. *In Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 254–265, 2002.

[107] B. Smith, W. Yurcik, and D. Doss. Ethical hacking: the security justification redux. *International Simposium on Technology and Society (ISTAS'02)*, pages 374–379, 2002.

[108] R. E. Stepp and R. S. Michalski. *Conceptual Clustering: Inventing Goal Oriented Classifications of Structured Objects*, pages 471–498. Morgan Kaufmann, Los Altos, CA, 1986.

[109] W. Stevens and G. Wright. *TCP/IP Illustrated Vol.2*. Addison-Wesley, 1994.

[110] C. Tarr and P. Kinsman. The validity of security risk assessment. *Proceedings of the 30th IEEE International Carnahan Conference on Security Technology (ICCST'02)*, pages 167–170, 1996.

[111] S. Theodoridis and K. Koutroubas. *Pattern Recognition*. Academic Press, 1999.

[112] H. T. Tian, L. S. Huang, Z. Zhou, and Y. L. Luo. Arm up administrators: Automated vulnerability management. *ispan*, 00:587, 2004.

[113] J. Wack and M. Tracey. DRAFT Guideline on Network Security Testing: Recommendations of the National Institute of Standards and Technology. *NIST special publication*, pages 800–842, 2001.

[114] W. Wang, J. Yang, and R.R. Muntz. STING: A statistical information grid approach to spatial data mining. In *The VLDB Journal*, pages 186–195, 1997.

[115] A. Whitaker and D. Newman. pages 37–45. Cisco Press, 2006.

[116] E. Yang, A. Erdogan, T. Arslan, and N. Barton. Multi-objective evolutionary optimizations of a space-based reconfigurable sensor network under hard constraints. In *Symp. on Bioinspired, Learning, and Int. Syst. for Security*, pages 72–75, 2007.

[117] C. Ying, A. Tsai, and H. Yu. Vulnerability assessment system (VAS). *Proceedings of the IEEE 37th. Annual 2003 International Carnahan Conference on Security Technology*, pages 414–421, 2003.

[118] I. Yoo. Visualizing windows executable viruses using self-organizing maps. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 82–89, 2004.

[119] S. Zanero. Analyzing tcp traffic patterns using self organizing maps. In *ICIAP'05:Proc. 13th International Conference on Image Analysis and Processing*, volume 3617, pages 83–90, 2005.

[120] S. Zanero and S. Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proc. of the 2004 ACM Symposium on Applied computing*, pages 412–419, 2004.

Appendix A

Notation

ACE - Application Characterization Environment
ACL - Access Control List
AI - Artificial Intelligence
AP - Access Point
ARFF - Attribute-Relation File Format
ATM - Asynchronous Transfer Mode
AU - Anti-Unification
BSI - British Standard Institute
CBR - Case Base Reasoning
CERT - Computer Emergency Response Team
CSI - Computing Services Center - *Centre de Serveis Informàtics*
CVE - Common Vulnerability Exposure
CVSS - Common Vulnerability Scoring System
DAU - Detailed Anti-Unification
DB - Davies-Boudin
DCSSI - *Direction Centrale de la Sécurité des Systèmes d'Information*
DHCP - Dynamic Host Configuration Protocol
DMZ - DeMilitarized Zone
DNS - Domain Name Service
DoS - Denial of Service
EALS - Enginyeria i Arquitectura La Salle
FP - Framework Program
FTP - File Transfer Protocol
GA - Genetic Algorithm
GP - Genetic Programming
GRSI - Group of Research in Inteligent Systems - *Grup de Recerca en Sistemes Intel·ligents*
IDS - Intrusion Detection System
IDWG - Intrusion Detection Exchange Format
IP - Internet Protocol
ISECOM - Institute for Security and Open Methodologies
ISMS - Information Security Management System
LAN - Local Area Network
LOPD - *Ley Orgánica de Protección de Datos de carácter personal*
LSSIC - *Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico*
MOCK - MultiObjective Clustering with automatic K-determination
MST - Minimum Spanning Tree
NN - Neural Network
NIST- National Institute of Standards and Technology
OCTAVE - Operationally Critical Threats, Assets and Vulnerability Evaluation
OS - Operating System
OSI - Open Systems Interconnection
OSSIM - Open Source Security Information Management
OSSTMM - Open Source Security Testing Methodology Manual
OSVDB - Open Source Vulnerability Database

PLC - Power Line Communications
QoS - Quality of Service
RF - Radio Frequency
SCP - Secure CoPy
SOM - Self-Organizing Maps
SSID - Service Set IDentifier
TCP - Transmission Control Protocol
UDP - User Datagram Protocol
URL - Ramon Llull University - *Universitat Ramon Llull*
VASE - Vulnerability Assessment Support Engine
WEP - Wired Equivalent Privacy
WLAN - Wireless Local Area Network