UNIVERSITY OF CALABRIA

DOCTORAL THESIS

Anomalies in cyber security: detection, prevention and simulation approaches

Author:
Luciano ARGENTO

Supervisor:
Prof. Angelo FURFARO
Co-supervisor:
Prof. Fabrizio ANGIULLI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

D.I.M.E.S.

June 1, 2018

Abstract

With the massive adoption of the Internet both our private and working life has drastically changed. The Internet has introduced new ways to communicate and complete every day tasks. Organisations of any kind have taken their activities online to achieve many advantages, e.g. commercial organisations can reach more customers with proper marketing. However, the Internet has also brought various drawbacks and one of these concerns cyber security issues. Whenever an entity (e.g. a person or company) connects to the Internet it immediately becomes a potential target of cyber threats, i.e. malicious activities that take place in cyberspace. Examples of cyber threats are theft of intellectual property and denial of service attacks. Many efforts have been spent to make the Internet perhaps the most revolutionary communication tool ever created, but unfortunately little has been done to design it in a secure fashion. Since the massive adoption of the Internet we have witnessed a huge number of threats, perpetrated by many different actors such as criminal organisations, disgruntled workers and even people with little expertise, thanks to the existence of attack toolkits. On top of that, cyber threats are constantly going through a steady evolution process and, as a consequence, they are getting more and more sophisticated. Nowadays, the cyber security landscape is in a critical condition. It is of utmost importance to keep up with the evolution of cyber threats in order to improve the state of cyber security. We need to adapt existing security solutions to the ever-changing security landscape and devise new ones when needed. The research activities presented in this thesis find their place in this complex scenario. We investigated significant cyber security problems, related to data analysis and anomaly detection, in different areas of research, which are: Hybrid Anomaly Detection Systems; Intrusion Detection Systems; Access Control Systems and Internet of Things.

Anomaly detection approaches are very relevant in the field of cyber security. Fraud and intrusion detection are well-known research areas where such approaches are very important. A lot of techniques have been devised, which can be categorised in anomaly and signature based detection techniques. Researchers have also spent much effort on a third category of detection techniques, i.e. hybrid anomaly detection, which combine the two former approaches in order to obtain better detection performances. Towards this direction, we designed a generic framework, called *HALF*, whose goal is to accommodate multiple mining algorithms of a specific domain and provide a flexible and more effective detection capability. *HALF* can be easily employed in different application domains such as intrusion detection and steganalysis due to its generality and the support provided for the data analysis process. We analysed two case studies in order to show how *HALF* can be exploited in practice to implement a Network Intrusion Detection System and a Steganalysis tool.

The concept of anomaly is a core element of the research activity conducted in the context of intrusion detection, where an intrusion can be seen as an anomalous activity that might represent a threat to a network or system. Intrusion detection systems constitute a very important class of security tools which have become an invaluable defence wall against cyber threats. In this thesis we present two research

results that stem from issues related to IDSs that resort to the n-grams technique. The starting point of our first contribution is the threat posed by content-based attacks. Their goal is to deliver malicious content to a service in order to exploit its vulnerabilities. This type of attacks has been causing serious damages to both people and organisations over these years. Some of these attacks may exploit web application vulnerabilities to achieve goals such as data theft and privilege escalation, which may lead to enormous financial loss for the victim. IDSs that exploit the n-gram technique have proven to be very effective against this category of cyber threats. However, n-grams may not be sufficient to build reliable models that describe normal and/or malicious traffic. In addition, the presence of an adversarial attacker is not properly addressed by the existing solutions. We devised a novel anomaly-based intrusion detection technique, called PCkAD to detect content-based attacks threatening application level protocols. PCkAD models legitimate traffic on the basis of the spatial distribution of the $n-$grams occurring in the relevant content of normal traffic and has been designed to be resistant to blending evasion techniques. Indeed, we demonstrate that evading is an intrinsically difficult problem. The experiments conducted to evaluate PCkAD show that it achieves state of the art performances in real attack scenarios and that it performs well against blending attacks. The second contribution concerning intrusion detection investigates issues that may be brought by the employment of the n-gram technique. Many approaches using n-grams have been proposed in literature which typically exploit high order n-grams to achieve good performance. However, because the n-gram domain grows exponentially with respect to the n-gram size, significant issues may arise, from the generation of huge models to overfitting. We present an approach aimed to reduce the size of n-gram-based models, which is able build models that contain only a fraction of the original n-grams with little impact on the detection accuracy. The reported experiments, conducted on a real word dataset, show promising results.

The research concerning access control systems focused on anomalies that represent attempts of exceeding or misusing access controls to negatively affect the confidentiality, integrity or availability of a target information system. Access control systems are nowadays the first line of defence of modern computing systems. However, their intrinsic static nature hinders autonomously refinement of access rules and adaptation to emerging needs. Advanced attributed-based systems still rely on mainly manual administration approaches and are not effective on preventing insider threat exploiting granted access rights. We introduce a machine learning approach to refine attribute-based access control policies based on behavioural patterns of users' access to resources. The designed system tailors a learning algorithm upon the decision tree solutions. We analysed a case study and conducted an experiment to show the effectiveness of the system.

IoT is the last topic of interest in the present thesis. IoT is showing the potential for impacting several domains, ranging from personal to enterprise environments. IoT applications are designed to improve most aspects of both business and citizens' lives, however such emerging technology has become an attractive target for cybercriminals. A worrying security problem concerns the presence of many smart devices that have security holes. Researchers are investing their efforts in the evaluation of security properties. Following this direction, we show that it is possible to effectively assess cyber security scenarios involving IoT settings by combining novel virtual environments, agent-based simulation and real devices and then achieving a means that helps prevent anomalous actions from taking advantage of security holes for malicious purposes. We demonstrate the effectiveness of the approach through a case study regarding a typical smart home setting.

Acknowledgements

First of all, I would like to express my sincere appreciation to my advisors Prof. Fabrizio Angiulli and Prof. Angelo Furfaro, for their precious guidance and support throughout my whole PhD study.

I would like to thank my colleagues for the time we spent together. Many thanks also to the co-authors of the publications referred to in this thesis.

I would also like to thank my friends and relatives, for their support and presence. Finally, my most heartful thanks go to my family, whose love, invaluable support and patience helped me sustain all the difficulties that I encountered during these three years.

Contents

List of Figures

List of Tables

Dedicated to my beloved family.

Chapter 1

Introduction

The Internet has revolutionised the way in which we stay connected with other people and perform our tasks, both in our private and working life, on a daily basis. It is undeniable that the Internet has become an essential part of our life due to the benefits that it carries with it. In this age of steep and fast technological advancement, we can complete many tasks easily and conveniently with just few clicks, such as paying our bills online, getting information about a topic of our interest, purchasing almost anything that we need, even food, by going through various websites. When it comes to business, the Internet has also brought a revolution, from recruiting employees to improving efficiency and productivity. Commercial organizations, for example, regardless of their size, invest a lot in Web applications supporting their activities. By taking their business online, such organizations achieve many advantages, e.g. they can reach more customers with proper marketing and they can cut costs in almost every aspect of their business.

Of course, like anything else, the Internet has also brought various drawbacks, such as addiction, trolling, bullying, health issues and so on. Among all these drawbacks, in this thesis we will focus on those concerning cyber security issues. As soon as any entity (e.g. a person or a company) starts using any Internet service it automatically expose itself to a wide variety of cyber threats, i.e. malicious activities that can occur through cyberspace, such as theft of intellectual property, destructive malware, denial of service attacks. These threats, often underestimated, have been causing minor to serious damages to both people and companies by affecting, for example, digital services and this also much likely happens on a daily basis [8].

Internet may represent, perhaps, the most revolutionary communication tool ever created but, unfortunately, it was not designed to be secure. Decades ago, when the Internet was going through its early stages of evolution, those who devoted themselves to its design and development were focused on technical issues with the aim of being able to transfer information quickly and reliably. These people were shortsighted about information security, they primarily took into account military threats, but they failed to forecast that the same Internet users a day might become threats. This led to a situation in which it was necessary to introduce ex post information security solutions in response to a very high number of threats spanning a wide range of severity.

While in the early years of the Internet there were few threats, the majority of which were perpetrated by crackers, since its massive adoption we have witnessed a huge number of threats. Nowadays, cyber threats are committed by not only crackers, but also other figures, such as criminal organisations and even people with little expertise. The latter category of attackers was born thanks to the development and distribution of attack toolkits. These toolkits, also known as crimeware, are bundles of malicious code designed to facilitate the launch of more or less sophisticated attacks on networked computers, e.g. the theft of sensitive information or a denial of

service (DoS) [155]. These kits, along with stolen information, are advertised and sold by cybercriminals in a black market of servers and forums [47]. An example of a well-known exploit kit is the Angler kit, which, according to [197] and [198], had been the most used kit between 2015 and 2016 before its creators were arrested [53].

On top of that, cyber threats are constantly evolving and becoming more and more sophisticated, attack after attack. Organisations, both private and governmental, have been struggling to keep up with the security of their systems and actually many of them have failed to meet the challenges posed by the existing threats. One of the reasons behind this failure is the lack of an adequate budget to address cyber security issues, which may be caused by the fact that the privacy and security of customer and business data were not valued enough. As a result, those organisations may face disruptive consequences, including huge financial loss, theft of customer data (as recently experienced by Yahoo [176], which revealed the largest incident ever reported) or, in the worst case, they might go out of business. A study conducted by the National cyber Security Alliance [137], published on April 22th 2015, highlighted that over 60% of all the small companies hit by cyber threats went out of business. In the light of the above discussion, it is clear how critical the cyber security landscape is nowadays. A lot of efforts are required in order to improve the state of cyber security and keep up with the evolution of cyber threats, and that means adapting existing security solutions in response to the ever-changing security landscape and devising new solutions when needed.

The research activities that we present in this thesis find their place in this complex scenario. As further discussed in the following section, we investigated relevant cyber security problems concerning the following topics: Hybrid Anomaly Detection Systems; Intrusion Detection Systems; Internet of Things and Access Control Systems.

1.1 Thesis contributions

The main objective of the present thesis is to improve different aspects of cyber security, from a data analysis perspective, with particular emphasis on the concept of anomaly.

We first present the research activity which seeks to improve hybrid anomaly detection systems to enable the combination of almost any existing technique, with the ultimate goal of making the analysis of anomalies more effective, in Chapter 3. In Chapters 4 and 5 we discuss two research activities conducted in the context of intrusion detection systems, where an anomaly can be seen as an anomalous activity which aims at compromising the target system or network. We considered the class of *content-based attacks* as intrusions and worked to address significant downsides of n-gram-based classification approaches, which tend to be very effective against the mentioned type of attack. We also devised an n-gram-based IDS technique that can stand against an adversarial attacker. We introduce a machine learning approach in Chapter 6 to improve the usage of access control systems in highly dynamic environments so to address anomalous activities that attempt to exceed or misuse access controls to carry out an attack against a target system. At last, in Chapter 7 we talk about the research we conducted to improve the assessment of security properties of IoT devices and to ultimately prevent anomalous actions from taking advantage of security holes for malicious purposes.

Overall, the following topics were the subject of the research activities conducted: Hybrid Anomaly Detection Systems, Intrusion Detection Systems, Access Control

Systems and Internet of Things. For each topic there is a section that explains the problem that has been considered, followed by the thesis contributions and motivations.

1.1.1 Hybrid Anomaly Detection Systems

The presence of anomalies in collected information, i.e. data that deviates substantially from what is normally expected, is a valuable source of knowledge and its discovery has many practical applications. One of the areas where anomaly detection is a very important problem is cyber security. Well-known examples of security-related problems that involve anomaly detection in such field are fraud detection and intrusion detection. The approaches that have been proposed to address anomaly detection fall in two categories: anomaly and signature-based detection. Anomaly-detection approaches rely on building models that suitably describe data patterns deemed as normal, however they may incur in the generation of a considerable amount of false positives. Signature-based techniques, which exploit a prior knowledge base of anomalous patterns, are able to effectively detect them but fail in identifying anomalies which did not occur previously. A lot of research efforts have been spent on another category of detection techniques, i.e. hybrid anomaly detection, which combine the two former approaches in order to obtain better detection performances. State-of-the-art hybrid approaches are typically bound to existing anomaly detection techniques by design and allow the analysis of only specific types of data. These constraints might limit the effectiveness of the analysis, in that it would be not possible to combine any existing anomaly detection techniques based on the capabilities they feature.

We designed a framework, called *HALF*, that allows to develop hybrid systems by combining available techniques, both signature and anomaly-based. The idea behind *HALF* is similar to the concept of defence-in-depth, which is about deploying multiple security tools with complementary capabilities together, to improve the security of the target system. *HALF* allows the deployment of any anomaly-based and/or signature-based techniques, thus it is possible to combine techniques with complementary capabilities to empower the anomaly detection analysis. *HALF* is here presented in a context where cyber security is the main topic, nonetheless our framework is not bound to any specific field of application, in fact it is able to operate on any data type. *HALF* has been designed to accommodate multiple mining algorithms by organizing them in a hierarchical structure in order to offer an higher and flexible detection capability. The framework effectiveness is demonstrated through two case studies concerning a network intrusion detection system and a steganography hunting system.

1.1.2 Intrusion detection systems

The concept of anomaly introduced in the previous section is very important in the context of intrusion detection. An intrusion can be seen as an anomalous activity that might represent a threat to a network or system. In order to detect such activities a very important class of security tools are employed, the so called Intrusion Detection System (IDS). IDSs have proven to be a valuable defence wall against cyber threats; they provide a means for monitoring networks and/or systems in order to detect activities which are potential or real violations of security policies. As a consequence, it is possible to gain a greater level of awareness about what is happening within a

network and/or system. So, suitable countermeasures can be taken against identified threats and, in addition, unknown vulnerabilities can be discovered as soon as possible.

There exist different classes of IDSs, determined by a variety of criteria such as the way an IDS is deployed, the nature of the algorithms that are used, etc. We focused on issues concerning anomaly-based IDSs [21] that resort to the *n-gram technique* to distinguish between anomalous and normal data units. An n-gram is a sequence of symbols that is extracted from a given input flow by using a sliding window of length n. This research activity consists of two works. The first work concerns the design and development of a novel n-gram-based intrusion detection technique and it is introduced in the following section. The second work follows the first one and it takes into account a variety of problems that derive from employing the n-grams to build classification models. The latter work is introduced in Section 1.1.2.

A n-gram-based intrusion detection technique

Anomaly-based IDSs that employ the n-gram technique have achieved satisfactory results in detecting intrusions, over the years. This category of IDS is very important, especially due to the existence of a class of attacks, called *content-based attacks*, which has been causing serious damages, year after year (e.g. see the `shellshock` attack [157]). However, counting on the n-grams alone might not be sufficient to create a reliable intrusion detection system. Indeed, n-grams that characterise normal data units may also occur in anomalous units and could make more difficult the task of detecting such anomalies. This aspect is not properly addressed by existing solutions, indeed they tend to focus on the detection of malicious n-grams.

Another very important aspect to consider is that nowadays in order to deploy an IDS in an organization network, it must be able to cope with the presence of an adversary attacker, i.e. a malicious user who aims at hindering the normal activities performed by the security tool. A well-known example of adversary attack is called *evasion attack*. Unfortunately, in literature many works do not address this aspect with sufficient efforts while others do not take it into account at all.

We address the above two issues by proposing *Packet Chunk Anomaly Detector* or *PCkAD*, a novel anomaly-based intrusion detection technique designed to monitor network traffic. Its uniqueness is to learn legitimate payloads by splitting packets in chunks and determining the within packet distribution of n-grams. This strategy is resistant to evasion techniques as blending. We prove that finding the right legitimate content is NP-hard in the presence of chunks. Moreover, it improves the false positive rate for a given detection rate with respect to the case where the spatial information is not considered. Comparison with well-know IDSs using n-grams highlights that PCkAD achieves state of the art performances.

Using compressed n-gram-based models for intrusion detection

In order to make n-gram-based techniques more scalable and reliable, an important concern is to keep under control the size of the achieved models. Generally, detection algorithms that resort to the n-grams achieve good performances by using high order n-grams, i.e. n-grams with $n \geq 2$. In effect, often even 2-grams are not sufficient to obtain good results. The main implication behind the usage of higher-order n-grams is that the n-gram domain grows exponentially with respect to the n-gram

size. In the context of network intrusion detection, where a n-gram symbol is represented by a byte value, with just $n = 3$ the total number of n-grams that can be observed is over 16 million. This aspect may lead to significant issues, from the generation of huge models to overfitting. In order to address these issues, we developed an approach aimed to reduce the size of n-gram-based models, which is able achieve models that contain only a fraction of the original n-grams with little impact on the detection accuracy. The experimental results, conducted on a real world dataset, show promising results.

1.1.3 Access control systems

Nowadays our daily activities rely on computing systems. The key element of modern computing systems is the data: controlling accesses to data is indeed of paramount importance. Access control systems are the first line of defence for data. These systems enforce fine-grained conditions which determine the users that can act. Multiple approaches to access control have been proposed, each of which enjoys different property. A common aspect of the existing approaches is that they are not meant to adapt over time according to new access needs and behaviours. For this reason, the policies employed at a given time instant encode knowledge available only at design phase. This lack of adaptation is a very important problem, because access control systems work in highly dynamic environments, therefore they get weaker and weaker over time. As a consequence, insider threats may arise, such as disclosure or stealing of sensitive data. Similarly to the previous works, this research activity focused on the concept of anomaly, which this time concerns the attempt of exceeding or misusing access controls to negatively affect the confidentiality, integrity or availability of a target information system. Continuative maintenance is required to preserve adequate access rights to new controlled resources and for changing users' patterns. We then propose a machine learning solution, called ML-AC, to address the above mentioned issues. ML-AC was designed to dynamically refine and update policies, respectively, to prevent insider threats and to automate policy administration. The capabilities of ML-AC are shown by means of a case study.

1.1.4 Internet of things

The latter contribution concerns the Internet of Things, which has been widely recognised as the next main step in the evolution of Internet and has rapidly become a synonym of opportunities for enterprises. There is a lot of hype around IoT, which is also backed up by outstanding predictions. According to Gartner, by 2020 the economic benefit brought by IoT will be close to $2 trillion. IoT carries a lot of significant benefits with it, such as sensor-driven decision analytics; process optimisation and optimised resource consumption. Of course, IoT is also impacting our private life, as demonstrated by its applications in smart homes.

When it comes to cyber security concerns, the situation is worrying. As reported by [8], by 2016 cybersecurity risks have received little attention. Most of the attention has been dedicated to the technologies needed to achieve the desired functionalities, neglecting the aftermath of the security issues that are going to arise. This has led to the production and deployment of insecure smart devices which have become juicy targets of cyber criminals [71]. With respect to traditional Internet sources of information, in the IoT scenarios data come from the physical world through the sensors installed on smart devices, thus widening the range of possible applications, e.g. involving the processing of environmental data and make intelligent decisions on the

surrounding environment. IoT has become a bridge between the physical and the digital world by including smart objects which interact with the physical environment without direct human intervention. This connection between the physical and digital world has brought to us new threat scenarios that were unthinkable years ago. On the positive side of things, recently there has been more interest in the existing security issues, starting from the growing attention received from governments [65] to the announcement of bug bounty programs for the discovery of vulnerabilities.

Researchers are trying to address the existence of vulnerable IoT devices by focusing on the evaluation of security properties. The goal is to identify and understand the security issues of currently deployed devices and provide guidelines and recommendations to manufacturers to help them fix the vulnerabilities, if possible, and improve the security of future devices.

To this end, we believe that virtual environments could be very helpful to assess security properties and discover vulnerabilities of IoT devices, in realistic scenarios. We then evaluated the effectiveness of a platform, called SMALLWORLD, which was designed to develop intelligent virtual environments where malicious and legal behaviours are simulated by means of the agent paradigm.

With this contribution we propose a solution that on one hand helps to mitigate the presence of security holes in IoT devices to prevent anomalous actions from taking advantage of them for malicious purposes. On the other hand, the solution allows the collection of data that can be analysed to evaluate the security properties of the devices of interest.

1.2 Thesis outline

The rest of the thesis is organized as follows. Chapter 2 introduces concepts that are crucial to the work presented in this thesis. Chapter 3 describes *HALF*, a framework designed to address anomaly detection in multiple domains by combining techniques that are both signature and anomaly-based. In chapter 4 we present PCkAD a novel anomaly-based intrusion detection technique thought to monitor network traffic and consider the presence of an adversary who tries to evade the detection. Chapter 5 presents an approach for compressing n-gram-based models, in the context of intrusion detection. Chapter 6 discusses a machine learning empowered access control system called ML-AC. Chapter 7 shows how virtual environments can be used to assess security properties and discover vulnerabilities of IoT devices, in realistic scenarios. Finally, Chapter 8 summarises the contributions, draws the thesis conclusions and analyses the future work.

Chapter 2

Background

In this chapter, the concepts that are fundamental background to the work presented in this dissertation are introduced. The chapter starts with a presentation of computer security, its definition, history and implications. Afterwards, machine learning and adversarial environment are covered.

2.1 Computer security

Computer security is defined in [184] as the collection of tools designed to protect data stored on the computer and to thwart hackers with malicious intentions. The previous definition mainly takes into account technology aspects. However, technology alone is not sufficient to address all the existing cyber threats that besiege organizations. Malicious users do not only target devices but also humans, e.g. by means of social engineering, and physical objects. In order to accomplish effective cyber defence an holistic approach is needed. The approach incorporates technical, human and physical elements to detect, prevent and correct cyber security vulnerabilities. Computer security have been a problem since the very beginning. In the 1970s computers were mainly used in academia, government agencies and large enterprises. Most of them were off-line, therefore they were hardly exposed to any malicious users, except for insiders. Very few significant cyber threats occurred in the 70s, indeed at the beginning most of the issues were caused by hacking. People behind these "threats" were guided by curiosity, the desire of having fun in breaking into systems and building reputations. The real cyber threats find their origin in cracking, which is the act of breaking into computers for criminal gain. In the 1980s the number of cyber threats started to grow up considerably. By the late 80s, the adoption of networks was growing rapidly and universities, militaries, and governments were connecting. As a consequence, the security need started to grow considerably as more and more cyber threats arose. Remarkable examples of such threats are the Vienna virus [72] and the Morris worm [164]. In particular, the latter was devised in 1988, and is a well-known worm that managed to close down much of the internet. In truth, the original goal of the Morris worm was to propagate on other people's systems stealthily, however, due to a critical mistake, the worm turned into a denial of service attack. By the middle of the 90s, network security threats had increased exponentially. This situation led to the creation of firewall and antivirus programs, security tools that aim at protecting computers from worms and viruses, and their mass production. The threats were so expansive that it was no longer possible to handle them by resorting to custom teams and measures. The internet has exploded since its massive adoption in the 90s and the number of malicious users, both amateur and professional, has also worryingly increased at an alarming rate.

In today's highly complex Internet environment, it is no longer sufficient to protect an organisation's network, even though it is still of utmost importance. With the emergence of applications accessible through the Internet, such as emails, Web services, File Transfer Protocol (FTP) and sales force automation systems, more and more people have started to conduct their every day business online, share all kinds of information through social networking. Obviously, this has caught the attention of users with malicious intentions. Unfortunately, these applications represent a juicy attack surface, in fact, due to their complexity they contain inherent vulnerabilities that can be exploited with enough knowledge and/or the right tools, for malicious purposes. This scenario has led to the evolution of network attacks into application-level attacks which have arisen as a new class of threats. Such threats requires more sophisticated defensive measures than those employed to protect systems against network threats. Over the years, industry has been paying increased attention to the security of applications (especially web applications) themselves in addition to the security of the underlying computer network and operating systems. Misuse of applications and Web services can result in the loss of valuable resources and millions of dollars.

Application level attacks will be covered in depth in a subsequent section, due to their importance in our work.

2.1.1 Vulnerabilities

In computer security, a vulnerability refers to a weakness in a system that exposes information security to a cyber threat. Attacker takes advantage of these weaknesses for a number of different reasons, from gaining financial information to turning the target system itself into a bot *schiller2011botnets*. In particular, software vulnerabilities are bugs (flaws) in applications (e.g., Web servers and browsers) that can be leveraged to make the application act in a way that it is not intended to. The attacker might deliver a piece of malicious code to the target system and run it in order to start downloading malicious software from the Internet and crashing the application. To exploit a vulnerability, sometimes deep knowledge on the system may be required, combined with custom malicious software, while other times, unfortunately, prefab malicious tools are sufficient. The latter approach, if available, allows even people with little expertise to accomplish malicious goals. In this frame, vulnerability is also known as the attack surface.

According to [48], the majority of software vulnerabilities typically stem from defects, bugs, and logic flaws, introduced by a relatively small number of common software programming errors. Many words have been spent to produce guidelines for software developers to help them develop secure coding best practices for their daily development work. However, despite the efforts, attackers are still exploiting old and new vulnerabilities in software. It might not be easy, but security must be an integral part of the software development. It may never be possible to get rid of all code defects, especially in very complex systems, but by following secure coding best practices developers can diminish the impact and frequency of software vulnerabilities.

Figure 2.1 shows the percentage of applications (mobile and non-mobile) that exhibited a certain type of vulnerability in 2015 [8]. Three of the most prominent examples of application vulnerabilities are Input Validation and Representation, Code Quality and API Abuse. Input validation is a security measure that is implemented to ensure that information systems only process well-formed data, preventing malformed data from triggering malfunctions and/or persisting in the database. Code

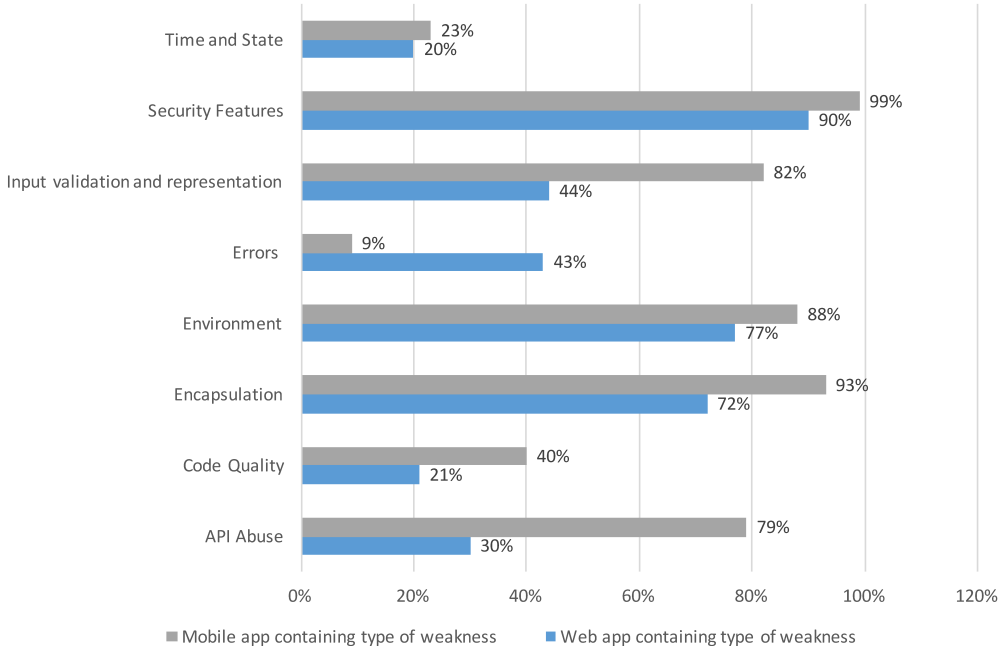


FIGURE 2.1: Type of weaknesses in mobile and non-mobile applications. The image has been taken from [8].

quality issues raise for not adopting programming best practices, which may lead to unreleased resources such as sockets and databases, and dereferences of null values. An API is a contract between two participants, i.e. a caller and a callee. The participant that typically tend to fail to honor the contract, leading to API abuse, is the former one. For example, consider a system that is exposed to the Internet. The admin should do his best to limit the damage whenever the system should be compromised. An effective method to accomplish this goal is the use of a *chroot jail*. This method aims at creating a temporary root directory for a running process so that it has a restricted view of the filesystem. However, if a program fails to call *chdir()* after calling *chroot()*, it violates the contract that specifies how to change the active root directory in a secure fashion.

2.1.2 Exploits

An exploit is a piece of software, e.g. a sequence of malicious instructions or a chunk of data, created expressly to take advantage of one or more vulnerabilities in a system in order to make it perform unintended behaviour.

The target of an exploit might be a specific organisation or a population of users, e.g. for stealing sensitive data and compromising as many hosts as possible to build a botnet *schiller2011botnets*, respectively. There exist many different vehicles and technologies behind delivery mechanisms, which are often characterised also by social engineering elements. With social engineering techniques attackers might fool employees and individuals into handing over valuable information or open a malicious file.

Many exploits are designed to get superuser-level privileges on a computer system. However, this goal might not be achievable by resorting to just one exploit, indeed it is not uncommon to witness an attack consisting of several exploits, first to gain low-level access, then to escalate privileges repeatedly until one reaches root. In the following a few examples of exploits are provided. *Unauthorized Data Access*

describes the scenario where someone/something accesses to arbitrary files and directories, without the right privileges, willingly or not. *Denial-of-Service attack* (DoS attack), such as the traditional ICMP and SYN floods, is a kind of attack whose objective is to consume all the resources of a target system (e.g. a server). We talk about *Distributed Denial-of-Service attack* (DDoS attack), when the DoS attack is perpetrated by multiple sources concurrently. An example of DDoS is to ask a server for information from multiple compromised user machines at the same time, this may result in a total system crash, corrupted services or a shutdown due to an enormous amount of allocated memory.

Application attacks

An application-layer attack targets the layer of the internet that is essentially intended for the end user. This layer is defined by the OSI (Open Systems Interconnection) model [61], a conceptual model that standardises and characterises the communication technologies of a communication system, and includes applications such as Google Docs, email, maps, weather and news, in other words everything we need in our daily lives. The application layer is perhaps the hardest to defend. The typical vulnerabilities of this layer are often defined by complex user input scenarios that are hard to encode into an intrusion detection signature. Moreover, the layer suffers from the highest degree of accessibility to the outside world. Indeed, the typical requirement for an application to function is that it must be accessible over Port 80 (HTTP [110]) or 443 (HTTPS [109]), two of the few ports that an organisation's firewall keeps open. These applications are referred to as web applications because the communications between them and the clients are based on the HTTP protocol.

As soon as a web application goes online, it immediately becomes vulnerable to attacks. The presence of vulnerabilities in web servers and programming flaws in web applications has led to an explosive increase in the number of web attacks over the past few years [111]. Such attacks can heavily affect organisations time and money and compromise their reputation. It is then paramount for an organisation to invest resources to build defence strategies and mechanisms.

Examples of common web attacks are briefly described in the following. *Arbitrary Code Execution* and *Code Injection* are two well-known examples of web application exploits. The former consists of leveraging a software bug in order to execute any commands of the attacker's choice on a target machine or in a process. The latter type of attack is intended to make a system processing invalid data to trigger a bug that allows the injection of malicious code into a target process to alter its execution flow . SQL/non-SQL injection and cross site scripting or XSS are two specific injection attacks. Cybercriminals use SQL and non-SQL injection against databases to export data such as Personally Identifying Data (PII), to delete accounts, create bogus accounts and modify data. A DoS can also be initiated. Cross Site Scripting or XSS is a variant on the injection attack. The main idea of XSS is to insert malicious JavaScript code to the back end web database. The insertion can be achieved by means of a blog comment post or a video. It is a very popular attack due to its capabilities of distributing malware, displaying illegitimate content or stealing session cookies and users login credentials

2.2 Machine learning

Machine learning (ML) is the field of study that, according to Arthur Samuel in [174], gives computers the ability to learn without being explicitly programmed. ML is a

sub area of artificial intelligence, which also intersects other fields of study such as statistics, theoretical computer science, mathematics and more. ML is about designing algorithms that can learn from data, known as training samples, and make predictions on future observations, known as test samples. The devised algorithms are meant to learn in an automatic way, without human intervention or assistance.

Machine learning is employed in a range of domains where designing and programming explicit algorithms with good performance is difficult or infeasible; indeed it has proven to be a very valuable tool and has found major applications in finance, healthcare, cyber security, robotics, and many more. Enterprises are benefiting from the use of ML algorithms and frameworks, thanks to their high predictive accuracy, in that they are now able to achieve company-wide strategies faster and more profitably than before. Machine learning algorithms can analyse enormous datasets and extract interesting information that can turn past enterprises' data into a competitive advantage and lead to strategic goals being accomplished, such as forecasting long-term customer loyalty, in reasonable time. All of this would be impossible to accomplish for a human due to memory constraints and obvious/redundant computations to perform for hours and days.

In the context of cyber security, ML finds numerous significant applications, and a major one is the exploitation of ML in intrusion detection systems for detecting malicious software or malicious network traffic. Conventional security software requires a lot of human effort to identify threats, extract characteristics from the threats, and encode the characteristics into software to detect the threats. This labor-intensive process can be more efficient by applying machine learning algorithms. ML algorithms can analyse large multidimensional data sets and identify anomalies, policy violations, signs of compromise and much more. As a result, a number of researchers have investigated various machine learning algorithms to detect attacks more efficiently and reliably.

There are three main types of ML approaches: unsupervised, semi-supervised, and supervised [44]. Before briefly explaining the mentioned approaches, it is important to introduce the concept of label. A label is a string or code that describes the nature of data instances or, in other words, it identifies the class a data instance belongs to. The label is generally the business or problem variable that experts assume has relation to the collected data. For instance, in the context of network intrusion detection, data instances (e.g. network packets) might be labelled with either *normal* or *anomalous*. In unsupervised learning problems, the main objective consists of finding patterns, structures, or knowledge in unlabeled data. When a portion of the available data is labelled the problem is called semi-supervised learning, while if all the data are labelled than the problem is called supervised learning. In the latter problem, generally the task to solve is to find a function that explains the data.

2.3 Adversarial machine learning

Advances in computing capabilities has made it possible to employ machine learning to solve a variety of tasks, many of which related to security, such as network intrusion detection [158] (e.g. to discriminate between malicious and legitimate network packets), spam filtering [161] (to discriminate between spam and ham emails), biometric identity recognition [32] (e.g. to discriminate between impostors and genuine users), in a practical way. However, when it comes to security domains, machine learning solutions that are solely designed to solve the target problem might not accomplish the desired results. Indeed, nowadays intelligent malicious users are

aware of the employment of machine learning as a tool to improve security. Such users, called adversaries or adversarial users/attackers, might go through a trial and error process to downgrade the performance of the machine learning system, e.g. by carefully crafting input data to figure out the weaknesses of the model used by the system. Adversarial users try to break many of the assumptions made by the authors of the machine learning solutions, e.g. data stationarity, which states that training and test data are drawn by the same distribution (even though it is typically unknown). This problem has led to the birth of a new area of research, namely Adversarial Machine Learning, whose goal is to design machine learning algorithms, able to hinder adversarial attacks, and to study the capabilities and limitations of adversarial users [104].

In the following we provide a description for the adversarial user and the taxonomy of attacks against machine learning based on [104]. An adversarial attacker can be described in three different ways, i.e. based on his capabilities, the type of security violation he can cause and the attacker's intention. The attacker may exhibit two capabilities: (a) the attacker is able to somehow alter the training data that is used to build the machine learning model (*a causative attack*) or (b) the attacker does not influence the learned model, but can send new data samples to the model and possibly observe its responses on these carefully crafted instances (*an exploratory attack*). The type of security violation the attacker may cause: either (a) making the machine learning system recognise harmful input as normal data (*an integrity violation*) ; (b) creating a denial of service event in which benign data samples are incorrectly filtered as false positives (*an availability violation*); or (c) using the system's responses to infer confidential information used in the learning process (*a privacy violation*). Concerning the attacker's intention, he may launch an attack with the aim of: (a) degrading the system's performance on one particular data sample or (b) causing the classifier to fail indiscriminately on a broad class of samples.

2.4 Summary

In this chapter we provided fundamental concepts for the understanding of the thesis. At the beginning of the chapter we defined computer security, which is the general theme of this thesis, and provided a brief discussion about its evolution. We highlighted that computer security is not just a collection of tools to improve the security of the a system, but it also incorporates human and physical elements to detect, prevent and correct cyber security vulnerabilities.

Vulnerabilities and exploits are two very important concepts related to cyber security, that were analysed in Section 2.1.1 and 2.1.2, respectively. These two concepts are very important to understand the works presented in Chapter 7 and 4, especially the former one.

In Section 2.1.2 we focused on a specific class of attacks called *application attacks*, i.e. attacks that target the layer of the Internet that is essentially intended for the end user. Application attacks include a class of attacks referred to as *content-based attacks*, which is explained in detail in Chapter 4 and constitute the objective of the intrusion detection technique presented in Chapter 4 and the classification approach presented in Chapter 5.

Machine learning is an important part of this thesis too. Indeed, ML is related to the works presented in Chapter 4, 5 and 6. As previously stated, when machine

learning is employed to develop a security tool or approach, it is important to consider the presence of an adversarial attacker. The research field of adversarial machine learning was discussed in Section 2.3 and has a very important role in the research presented in Chapter 4.

Chapter 3

A Hierarchical Hybrid Framework for Modelling Anomalous Behaviours

Anomaly detection refers to the problem of finding patterns in data that do not conform to what it is expected [46], that are mainly known as anomalies or outliers in different application domains. Anomaly detection is a common problem to many areas such as fraud detection [75], speech recognition [2], military surveillance for enemy activities [135], intrusion detection for cybersecurity [24] and detection of anomalies in astronomical data [171, 69]. This wide range of applications is due to the fact that very often anomalies are sources of significant or critical knowledge that can be derived from raw data. Examples of anomalies are banking transactions from unusual places which could indicate credit card or identity theft, in the context of fraud detection; presence of malformed strings in network packets, for cyber security; anomalous energy consumption, both for cybersecurity [123] and energy consumption [208], and so on.

The problem of anomaly detection has received a lot of attention since the beginning of the 19th century in the statistics community [70]. Recently, interesting advances in the application of machine learning algorithms for anomaly detection have been achieved. Over the years the research community has proposed a considerable variety of techniques, many of which are very specific to their application domains [29].

Signature-based techniques aim at analysing the data to find one or more matches with a set of rules or signatures. Signatures can include specific strings or regular expressions that characterize one or more classes of anomalies. These approach typically generates few false positives, however they fails to detect unknown anomalies or variants of known ones.

Anomaly-based techniques use suitable models that represent normal data and classify as anomalous data which deviate considerably from what expected by the model. Unlike signature-based techniques, they are able to discover both known and unknown anomalies. The main problem of these approaches is the generation of a not negligible amount of false positives.

Despite both classes of techniques have been widely used, the above issues have not yet been fully resolved. Some researchers have thus focused their attention on hybrid techniques, which are considered a viable solution. Such systems are designed to include the features of both signature-based and anomaly-based techniques in the attempt to gain the advantages of both of them and, at the same time, to mitigate their defects. Moreover, within the same class of techniques there exist many different approaches for the identification of anomalies, such as those based on statistics, machine learning and data mining for anomaly-based techniques. Each

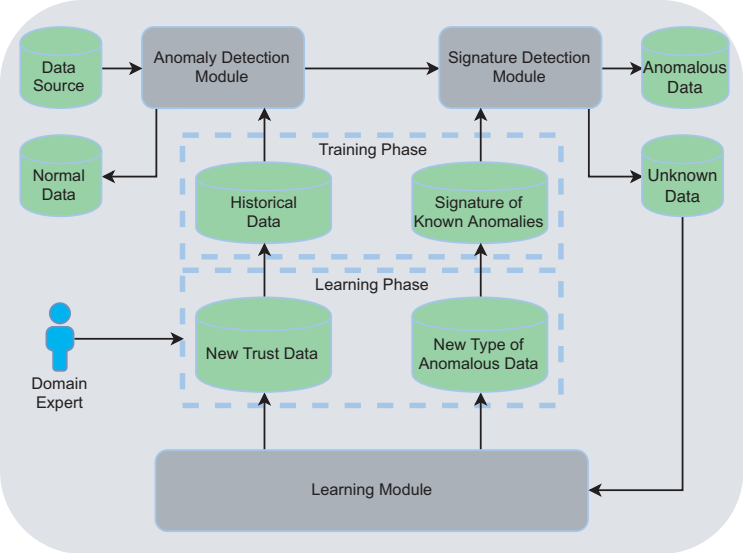


FIGURE 3.1: Framework Architecture

devised technique has unique features, strengths and drawbacks. Combining more of them together can lead to better results.

In this chapter we present a flexible multi-domain framework, called *HALF*, that generalizes the problem of anomaly detection. The framework is designed to embrace both signature-based and anomaly-based techniques. In addition, it makes possible to combine the use of different models for the analysis of data, organized in a hierarchical structure. Given its nature, *HALF* can work on any kind of data, unlike the existing works which are bound to specific fields of application.

The rest of the chapter is organized as follows. Section 3.1 summarizes the related work. Section 3.2 details the *HALF* architecture while Section 3.3 explains the design choices. Section 3.4 presents two case-studies and finally Section 3.5 summarises the work.

3.1 Related work

Hybrid approaches, subject of great interest in the research community, aim at exploiting the strengths of individual components, to obtain benefits from their combination. They are widely used in the domain of cyber security, especially for monitoring network activities, where many hybrid intrusion detection systems have been proposed [68].

A hybrid network intrusion detection system (HNIDS), which combines two anomaly techniques, i.e. packet header anomaly detection (*PHAD*) [146] and network traffic anomaly detection (*NETAD*) [145], with a misuse technique, was proposed in [22]. In particular, by using Snort [52] as the misuse engine, they exploited its pre-processors in order to integrate both *PHAD* and *NETAD*. The system resorts to the *libpcap* packet-capturing library to collect packets. Packets are first analysed in sequence by *PHAD*, then by *NETAD* and at last by Snort. The reported results show the effectiveness of this hybrid approach.

Another hybrid system is presented in [63]. It runs the anomaly and signature based analysis in parallel and then combines the results through a combination module. A SOM neural network is used to model the normal behaviour while the J48 classifier is exploited to classify known attacks. At last in order to interpret the

results three rules are defined: *(i)* if the modules are in agreement then the network packet is classified according to the output *(ii)* if the misuse module detects an attack then the packet is classified as such *(iii)* if an anomaly is detected while the misuse module does not detect any attack then the packet is classified as unknown attack.

Hybrid systems have not only been proposed in the context of network intrusion detection, but also in other sub domains of cyber security, including steganalysis. A novel approach for detecting concealed data in digital images has been reported in [150] where an anomaly-based approach, using hyper-dimensional geometric methods to model steganography-free images, is exploited instead of the widely used signature-based classifiers. The classification algorithm is also a hybrid signature-anomaly based technique when instances from multiple classes are available for training. A signature-based classifier is used to label instances when they are not enclosed by any class models. The authors focused primarily on JPEG images, but they stated that the proposed approach can also be applied on other types of files.

As discussed in the following, there are also other domains where hybrid approaches have been adopted.

Koeppen et al. [129] developed and evaluated an hybrid approach that analyses thermal infrared satellite time series data for detecting excess energy radiated from thermal anomalies such as active volcanoes. The authors aimed at addressing the limitations of MODVOLC [211], a state-of-the-art volcano monitoring system. The proposed a system is a combination of MODVOLC and RST [170] algorithms. Specifically, the system first processes the data by using MODVOLC. Subsequently the data undergo a timeseries analysis which has the objective of detecting anomalies that could not be identified by MODVOLC. This hybrid system too was designed to combine specific techniques. The authors demonstrated that it achieves better performance than MODVOLC alone, through an extensive experimental campaign.

Kumar et al. [133] propose the combined use of both data-driven and physics-of-failure models for fault diagnosis and life prediction, by developing an hybrid prognostics and health management methodology. Their goal is to combine the benefits of the two individual approaches, i.e. the possibility to define the healthy behaviour of the system by training and the ability to isolate the root causes and failure mechanisms that contribute to system failure, respectively. The results demonstrate the effectiveness of the approach proposed.

The architecture of *HALF* is also inspired by some characteristics of the human immune system [58], that include the learning component and the multi-layered defence mechanisms. In addition, the framework also provides other capabilities which distinguish the human immune systems such as signatures extraction mechanisms and storage. Artificial immune system are widely used in fields such as computer security (e.g. virus detection and process monitoring), anomaly detection, fault diagnosis and pattern recognition [58].

In [59] is presented a work that shares similar characteristics with our proposal. The author proposes a multi-agent system, inspired by the immunological principles for network intrusion detection. There are three types of agents: *i)* monitoring, *ii)* communicator and *iii)* decision/action. The first type of agents have the objective of searching for events of interest. The second type serves the purpose of carrying messages to enable cooperation among agents. In the context of the natural immune system, these agents would correspond to lymphokines, i.e. protein mediators secreted from T cells for stimulating B cells and antibodies. The third type of agents make decisions in response to detected events. The system takes into account multiple layers for the analysis and supports the detection of both known and unknown

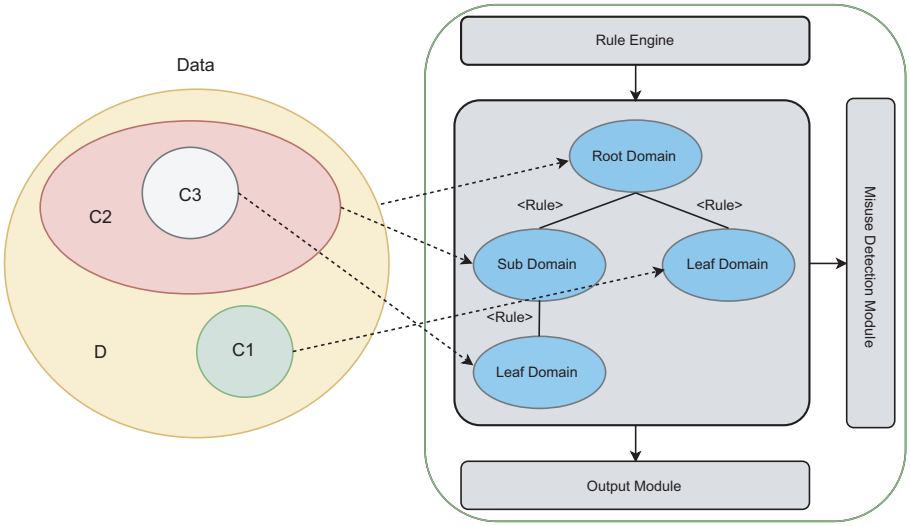


FIGURE 3.2: Anomaly Detection Module

intrusions. It monitors different parameters, from the packet level (the lowest) to the user level (the highest), and performs a correlation among them to support the detection of anomalous activities.

The approaches previously discussed suffer from a few limitations. They typically are bound to specific techniques. For example, in the case of [22] the hybrid system was designed to combine two anomaly-based (PHAD and NETAD) and a misuse-based (SNORT) techniques. Sometimes they are also limited to specific types of data, like the pcap format [22], while other times it is not clear what kind of data format is supported [63]. The solution presented in [59] has a completely different infrastructure with respect to the other ones, nonetheless it also lacks the support for the deployment of existing anomaly detection techniques.

3.2 Description of the framework

HALF has been designed to be applied to different domains, so it is not tied to any particular type of data. In light of the previous consideration, we will use the term *data instance* to indicate the basic unit of data processed by the detection techniques.

The framework architecture is shown in Figure 3.1 and consists of three main modules: *i)* anomaly detection, *ii)* signature detection and *iii)* learning. The role of the last module is to support the first two in detecting and handling the concept drift [207]. The anomaly and signature detection modules employ detection techniques, such as data mining techniques, which make use of different data sources, one containing historical data of interest and the other signatures. Below we explain the logical key steps for data instance processing.

The data instance is routed toward the anomaly detection module that marks it as normal or anomalous based on the previously observed data. If the data instance is marked as anomalous, then the misuse module checks whether it holds a known signature, and in the positive case notifies a matching. The data instances that do not match any known pattern could be either a new type of anomalous data or unseen normal data. In the last step, the learning module tries to establish if the data instances suggest the presence of a new concept with the intent to update the right data source, in order to maintain an updated view of the incoming data.

From a formal point of view, the anomaly detection module receives as input D, an instances-set which is composed by unlabelled normal and abnormal instances that we denote with N and A, respectively. N is given by the union of known (N_k) and unknown (N_u) normal instances, where N_k are documented instances. Instead A will be compounded by known (A_k) and unknown (A_u) abnormal data.

Before going into the details, we assume that all instances of A are interesting for the purposes of our research / study.

Below are reported the properties of the different types of data:

Property 1: $N \cup A = D$ and $N \cap A = \emptyset$.

Property 2: $N_k \cup N_u = N$ and $N_k \cap N_u = \emptyset$.

Property 3: $A_k \cup A_u = A$ and $A_k \cap A_u = \emptyset$.

The *Anomaly Detection Module* generates two output data streams, O_n and O_p. The former is composed of $N_k \cup \hat{A}_u$, where $\hat{A}_u \subset A_u$ represents the false negatives, i.e. abnormal data classified as normal, under the assumption that N_k is signature-free ($N_k \cap A_k = \emptyset$). Such stream does not require further processing. The aim of the *Anomaly Detection Module* is to make empty the set $\hat{A}_u$, a goal which is typically easy to achieve for the anomaly-based detection techniques. The latter stream contains $N_u \cup A_k \cup (A_u \setminus \hat{A}_u)$ and is directed to the *signature Detection Module*. The module provides $N_u \cup (A_u \setminus \hat{A}_u)$ as output, with the assumption that the false positive rate of a signature detection technique is very low. The task of the *learning Module* is to merge N_u with N_k and $(A_u \setminus \hat{A}_u)$ with A_k. This operation coincides with the online learning phase, detailed next. By looking the previous flow it is fundamental to minimize the false negatives $\hat{A}_u$ and to properly do unions of sets.

In the following each single component is explained in details and, at last, the initialization phase of the framework is discussed.

3.2.1 Anomaly Detection Module

The *Anomaly Detection Module* shown in Figure 3.2 has a hierarchical tree structure. The structure is composed by n levels, each of which contains one or more nodes, except for the root level. The idea at the basis of the hierarchical structure of the module is to create a parallelism with the possible presence of a hierarchical structure in the data source. Let us consider Figure 3.2, it shows the set of data D to be analysed by the module. Within D the subsets $C1$, $C2$ and $C3$ can be identified. Each of these subsets corresponds to a class of data which might require a specific detection technique in order to be properly analysed. To meet this need, each class is associated with a node of the hierarchical structure which receives instances of the specific class and hosts the required technique. A specific technique might also be deployed on two or more nodes, with different configurations, as different classes of data may be analysed by the same technique with a different setting.

In order to determine the destination node of the data that must be processed by an appropriate detection technique, the framework provides a rule engine. A user can utilise the engine to create rules in a semi-automatic way, indeed it only requires to specify the fields of interest of the data to be analysed. These fields are used to build a rule, by which the data of interest are filtered from the data stream and routed to the selected node. Rules are represented as boolean expressions, according to the grammar reported in Figure 3.3, allowing relational operators among variables, corresponding to data instance fields, and literals. Figure 3.4 shows an example of such a rule.

This approach is based on the *divide et impera* strategy: whenever you come across a data analysis problem, you might want to divide the problem into two or more simple sub-tasks, depending on the characteristics of the data. For example, in the context of cyber security, the detection of some types of network anomalies can be split into the simpler problems of detecting anomalies in network traffic of each host. Indeed, they may have a very different behaviour, making a global analysis very difficult or inaccurate. In the medical field, the nodes of a certain level may be associated with different age groups of a set of people. The same data may have a different meaning for each age group, for which a data instance is an anomaly for a certain age group, while it is normal for another group. The last example discussed highlights a specific type of anomaly, defined contextual anomaly. There are three different types of anomalies [46]: (i) point anomalies, a single data instance that can be considered as anomalous with respect to the rest of data. This is the simplest type of anomaly and is the focus of majority of research on anomaly detection; (ii) anomalous sequences, a sequence of data instances that is anomalous with respect to the entire dataset though the single data instances may not be anomalies by themselves; (iii) contextual anomalies, data instances that are anomalous in a specific context but not otherwise.

The contextual anomalies are managed through the use of rules and the hierarchical structure, as previously described, while the remaining two are dependent on the model of the techniques used. Care must be taken when seeking anomalous sequences within a node. Indeed, the creation of a new child could prevent his father to see part of the data that make up a behaviour. Therefore, when using a technique to detect anomalous sequences, it should be directly used in a leaf node, or should ask their children to forward to it the non-anomalous data.

Typically an anomaly detection technique can produce two types of output: scores and labels. The former represent the degree of certainty a given data instance or data instances is considered an anomaly. In the second case, the techniques assign a label, normal or anomalous. However, the framework does not place any restrictions on the type of output that an anomaly detection technique can generate.

3.2.2 Signature Detection Module

The signature detection module can be seen as specular with respect to the anomaly detection module, as represented in Figure 3.5.

Each node belonging to the anomaly detection structure might be linked with a node that employs a signature-based detection technique. If there exists a link, when a data instance is marked as anomalous, a signature node is probed to establish if the anomaly notified matches with any known signature or it is a potential false positive.

In the presence of techniques that work on signatures characterized by sequential events, the same considerations made for the anomaly detection module hold. It

⟨*rule*⟩ ::= ⟨*bterm*⟩ { '||' ⟨*bterm*⟩}
⟨*bterm*⟩ ::= ⟨*bfact*⟩ { '&&' ⟨*bfact*⟩}
⟨*bfact*⟩ ::= ⟨*fact*⟩ ⟨*relop*⟩ ⟨*fact*⟩ | '!' ⟨*fbact*⟩ | '(' ⟨*rule*⟩ ')'
⟨*fact*⟩ ::= var | boolv | numv | '' stringv ''
⟨*relop*⟩ ::= '<' | '<=' | '==' | '!=' | '>=' | '>'

FIGURE 3.3: EBNF syntax for rules

Data Type:	Name <string>	Age <num>	Role <string>	Enrolled <bool>
Valid rule:	Enrolled && (Age>= 18 && Age< 80)			

FIGURE 3.4: Example of a rule

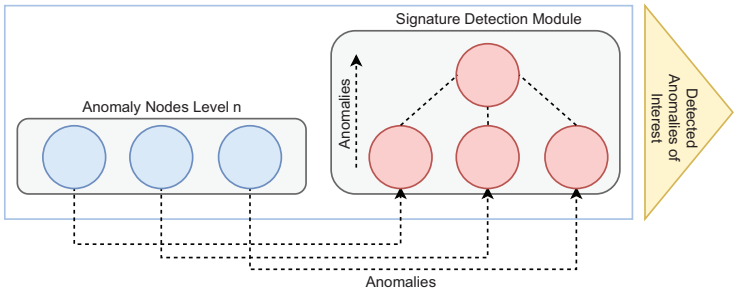


FIGURE 3.5: Signature Detection Module

may be necessary to forward some data along the hierarchical structure of the signature module. In the light of the above discussion, the output produced by the signature module is the set $A \setminus (\cup_{i \in S} A_i)$ where A_i denotes the subset of signatures associated to the i-th node and S the subset of nodes in the signature module that were asked to process the anomalies.

The purpose of a signature is to describe the characteristic elements of a specific anomaly or set of anomalies. A signature can have different forms, e.g. it could be a simple string or a rule. Generally, a signature is of good quality if it is narrow enough to precisely characterize a given set of anomalies, and, at the same time, it should be flexible enough to capture variations. If those requirements were not met, a high number of false positives or false negatives could be observed [131].

Usually, when a signature-based technique identifies a signature, it generates as output a label that describes the nature of the anomaly.

According to the application domain of the framework, the detected signatures may have a different meanings. For example, in the case of intrusion detection systems, signatures match patterns through which is possible to recognize potential attacks. In the area of steganography [150], instead, a signature identifies the presence of a hidden message inside an image through which terrorists, criminals, and other hostile entities can plan and coordinate their illicit activities.

Using signature-based modules has a considerable impact on the analysis conducted on the notifications generated by an anomaly-based technique. All known true positives are recognized and set aside, leaving only data that have never been seen to be further analysed or processed. In conclusion, the output of signature module will be $A_u \cup N_u$ under the assumption that the false negative rate of a signature detection technique is in general very low, such set of data instance there will be the input of the learning model, with the scope to discover new concepts or signatures.

3.2.3 Learning Module

The learning module is the most critical component of the framework and it is responsible for the evolution of the detection techniques. If a data instance reaches

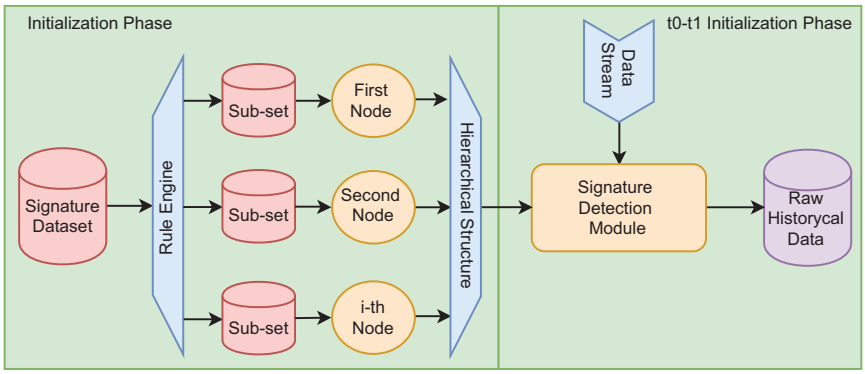


FIGURE 3.6: Initialization phase

this module, it belongs to the set $A_u \cup N_u$, that represent the set of data instances uncovered at the current time. Part of the set could be a consequence of data evolution. Indeed, in many domains the nature of data keeps evolving and a current notion of normal data might not be sufficiently representative in the future [46]. This phenomenon takes the name of concept drift [199] [101]. In order to learn and maintain an accurate detection level it is fundamental to detect the change and adapts the model to classify new data. Techniques developed to overcome concept drift can be divided into three categories: adaptive based [66] [45], learners which modify the training set [3] and ensemble techniques [201]. The framework provides support for the evolution for those techniques that do not natively support the concept drift.

Finding a new signature corresponding to an unknown anomaly is not a simple task because any assumption depends on the application domain. Using only prior knowledge may not be enough to establish if a new single or sequence of data instances can be an anomaly of interest. To handle effectively this task it is necessary the cooperation of a domain expert that can analyse the suspect data instances and build new signatures. The objective of this module is to simplify and reduce as much as possible the human work.

The learning module performs two tasks to accomplish his purpose:

- Scores top anomalies from $A_u \cup N_u$;
- Shows the data instances that more than others can be anomalous, based on the assigned scores, to a domain expert in order to discover unknown signatures and update the techniques hosted by the signature module.

The learning module operates by creating a temporary dataset for each node that requires support for the model evolution, in which new data instances in $A_u \cup N_u$, observed during a chosen temporal window, are stored. The framework builds clusters from these datasets, by using the models that need to be kept up to date. This strategy is necessary as the models are the only ones to know the features considered relevant for characterizing a data instance. Each cluster is then analysed by a domain expert / analyst who must determine whether it represents a set of new normal or abnormal data instances. In some cases, these operations can be automated, when strong assumptions about the data to be searched are available. Another domain expert's task is to establish a criterion, when performing the verticalization of the framework, according to which a cluster can be analysed or requires an additional process of construction.

Once the new instances have been labelled, the knowledge gained can be integrated into existing models in order to keep them updated. For this purpose, the

framework may employ a technique for managing the concept drift based on ensemble (or by modifying the training set), such as those proposed in [201] and [154].

Concept drift is a research area in its own right that requires a thorough study. In addition, its management involves a in-depth analysis of its implications in various domains. For these reasons we decided to not further address the topic here, but to investigate it in a future work.

3.2.4 Initialization phase

The initial training phase involves two aspects: the training of signature-based and anomaly-based models.

As regards the training of the signature-based models, it is assumed that the dataset of signatures is available. This assumption is almost always valid since the signatures represent what the analyst knows and wants to find. With this set of data, the signature module is trained and put into operation mode.

Training the anomaly module can result in a more complex task than the process previously described. In this case, a data set of normal data is not always available. Even if they were available, they may not be suitable for the analysis to be conducted. For example, in the fraud detection domain, given a dataset containing data instances about a certain group of subjects, if an analyst wants to analyse the behaviour of a different group of subjects, the information already available are not suitable for determining which transactions are normal and which are not. Therefore the analyst has to collect new data instances by observing the behaviour of the target subjects.

If the object of the analysis is a data flow, in order to obtain the dataset of normal data, the framework collects data for the necessary time only with the signature module enabled. A domain expert should analyse the resulting dataset, since it is possible that some unknown anomalies are collected. At the end of this preliminary phase the anomaly module can be trained. Figure 3.6 shows the training process discussed above.

3.3 Framework design

This section presents some details about the design of the framework, which has been implemented in the C++ language, and focuses on the most relevant aspects. The design process has been driven by the rigorous application of the software engineering principles, mainly modularity, and exploited several design patterns [84].

The UML class diagram reported in Figure 3.7 shows the main classes handling data instances and managing rules. They are employed to build rules used to filter and route data instances within the hierarchical structure. The abstract class `DataInstance` generalizes the notion of data instance and enables the framework to be used in different specific domains. A `DataInstance` object is a collection of pairs *(field, value)* allowing to access structured information from raw data representation, e.g. IP fields from the raw bytes of an IP packet. Given a raw data representation, the fields of data instance are defined by means of a JSON configuration file. Rules are modelled by the class `Rule` and are evaluated with respect to `DataInstance` objects. `Rule` objects are created by a suitable instance of the `RuleBuilder` class which uses the data instance type configuration file to define the building strategy and to parse a raw rule, expressed accordingly to the

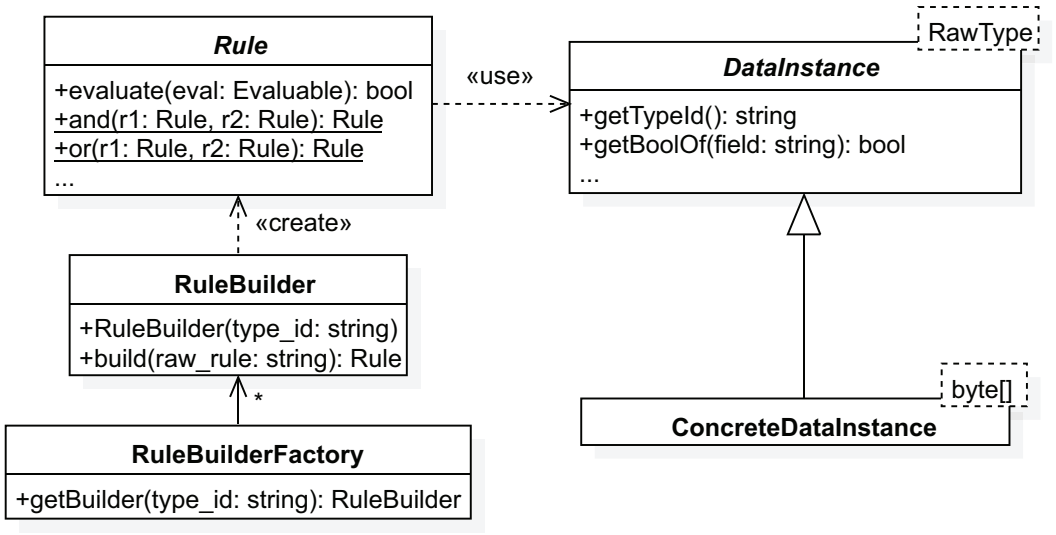


FIGURE 3.7: Rule handling classes

previously defined syntax. Finally, `RuleBuilderFactory` keeps an instance of `RuleBuilder` for each type of data instance allocated within the framework.

The class diagram reported in Figure 3.8 shows the classes introduced to represent the hierarchical structure of the mining modules and to embed data mining techniques. The base class `Node` introduces the data structures modelling the hierarchical relations. In particular, each `Node` object stores the references to its parent and children. In order to handle the routing of data instances between anomaly and misuse hierarchies, a node may keep a reference (in its `link` field) to a node belonging to the other tree. Each node is also equipped with a `Rule` instance which establishes the condition that has to be satisfied by a data instance for being suitable to be processed by at least one mining technique among those embedded in the subtree rooted at it. As described in the previous section, the most specific technique will be effectively employed for processing the data instance.

The `process` method defines how a data instance is processed and optionally routed to the other hierarchy.

The bound between a node and the relevant mining model is kept by the `MiningNode` class which specializes the base class `Node`. A mining model is represented by the class `Model` which introduces suitable methods defining the typical model handing functions such as: training, model updating and data instance classification.

Two specific subclasses of `Model` are defined, i.e. `ClassificationModel` and `RegressionModel`, depending respectively on whether the model is used for classification or regression. The `AnomalyNode` and `SignatureNode` classes have been introduced to distinguish between nodes which are part of the anomaly or signature hierarchy respectively.

The starting point to build a concrete application based on *HALF* is to create an instance of the `HALFApp` and then to feed it with the suitably configured detection hierarchies (anomaly and signature). The structures of these hierarchies and their interconnections are achieved by parsing a configuration file. The class diagram of Figure 3.9 shows the classes introduced to connect an *HALF* application with the data sources, to configure the routing strategies along and between the two hierarchies and to handle the training and evolution (concept drift) phases.

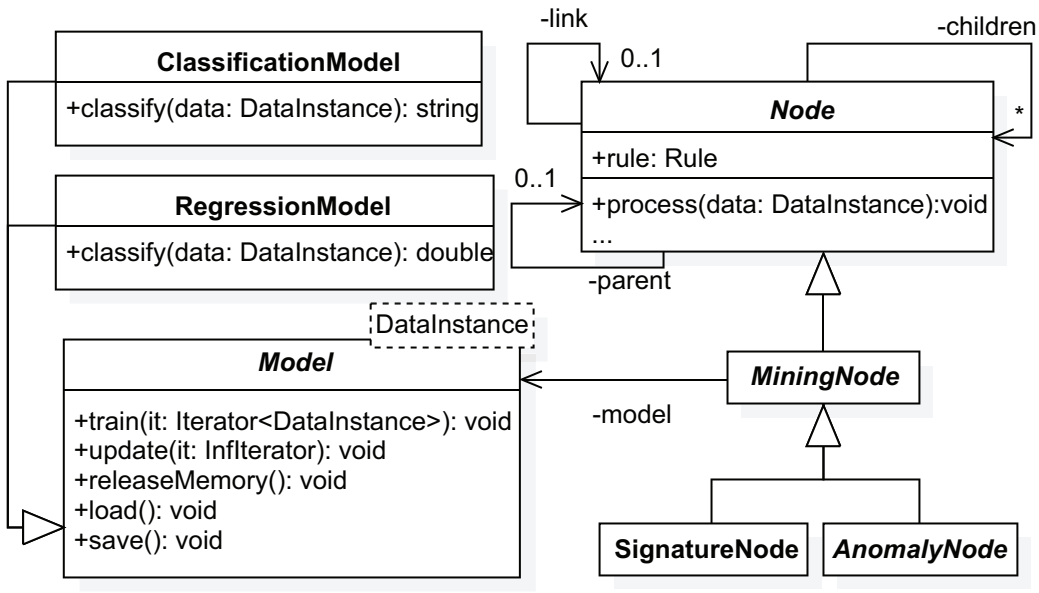


FIGURE 3.8: Hierarchy classes

An instance of the `InputSource` class has the task of extracting the data instances from a given data source and transform each of them into an instance of the a subclass `DataInstance`. The class `OutputSink` defines objects whose role is that of producing the output of the classification process in a suitable format, e.g. alerts or logs, according to what specified by the analyst during the configuration phase. The class *Router* embeds the routing strategy according to which actual data instances are delivered to target nodes, by exploiting the rules associated with them. By default, the framework provides a breadth-first strategy. The `LearningModule` class is in charge of handling training of the models (during the initial learning phase) and concept drift aspects (during the execution).

3.4 Case studies

This section presents two case studies that demonstrate how *HALF* can be exploited in practice for implementing a Network Intrusion Detection System and a Steganalysis tool, respectively.

3.4.1 Network Intrusion Detection System

In this first case study, the framework is verticalized for the analysis of network traffic. The first step is to determine the nature of the data instance. We consider packet-switched networks [91], therefore the data instance corresponds to a network packet which is a sequence of bytes containing control information and user data [78]. For the purposes of verticalization, the user must specify which are the fields that constitute a packet and how they should be extracted. In addition, a suitable module for reading input data has to be implemented. In this case, packets are read by a *sniffer* process that listens on a network interface [13]. The framework already offers a network package, which contains all the tools for reading and handling network packets in `pcap` format [105], so there is no need to implement what was discussed

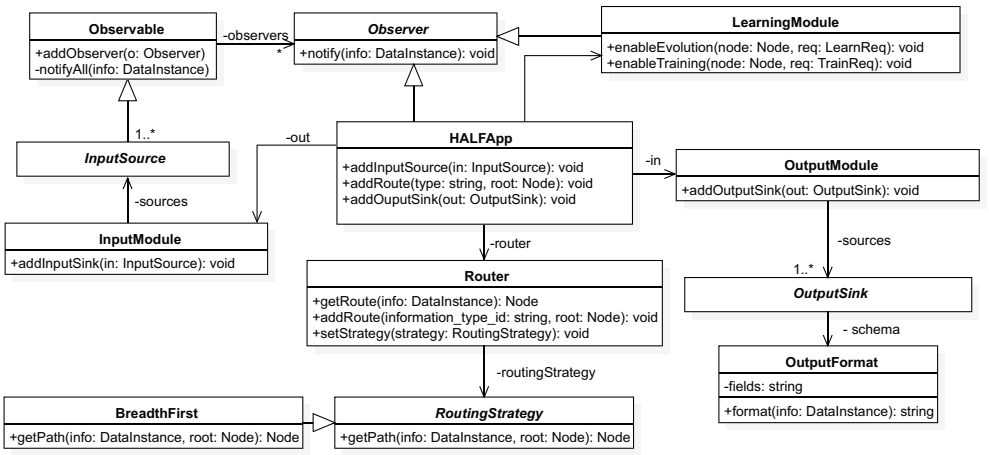


FIGURE 3.9: *HALF* application classes

so far. However, we want to point out that the steps described above are important to understand how to verticalize the framework in the chosen context of analysis.

The problem of anomaly detection in networks [136] is to find patterns in the traffic that does not conform to that considered as normal behaviour. The anomaly detection systems unfortunately tend to generate many false positives and for this reason they are poorly used within organizations networks. However they are most commonly employed in hybrid architectures [68], combined with a signature-based detection systems. The latter type of system takes the name of misuse-based, in this domain.

We chose two techniques for the detection of network traffic anomalies: PAYL [203] and PCkAD [11]. PAYL analyses network traffic payloads with the aim of building a byte-frequency distribution model, with 1-grams and unsupervised learning. It was designed to build a separate model for each combination of destination port and data instance length. It is able to process application-level protocols such as SMTP, HTTP and TELNET. The Mahalanobis distance is used to determine whether the current traffic is anomalous. PCkAD was designed to identify content-based attacks over HTTP and FTP protocols. It builds models of normality from network traffic to learn the spatial distribution of higher order n-grams. The goal of this technique is to detect unseen patterns and legitimate ones used in an anomalous way. These two techniques were compared in [11] and results show that PCkAD performs better than PAYL with respect to the type of traffic analysed, i.e. FTP.

SNORT [52], a well-known signature-based system, was chosen to integrate the anomaly-based techniques. Since SNORT is a stand-alone system which captures packets only through a network interface, it was necessary to develop a wrapper for its integration. The wrapper performs the following steps:

- creates a virtual network interface (i.e. `snort_tap`);

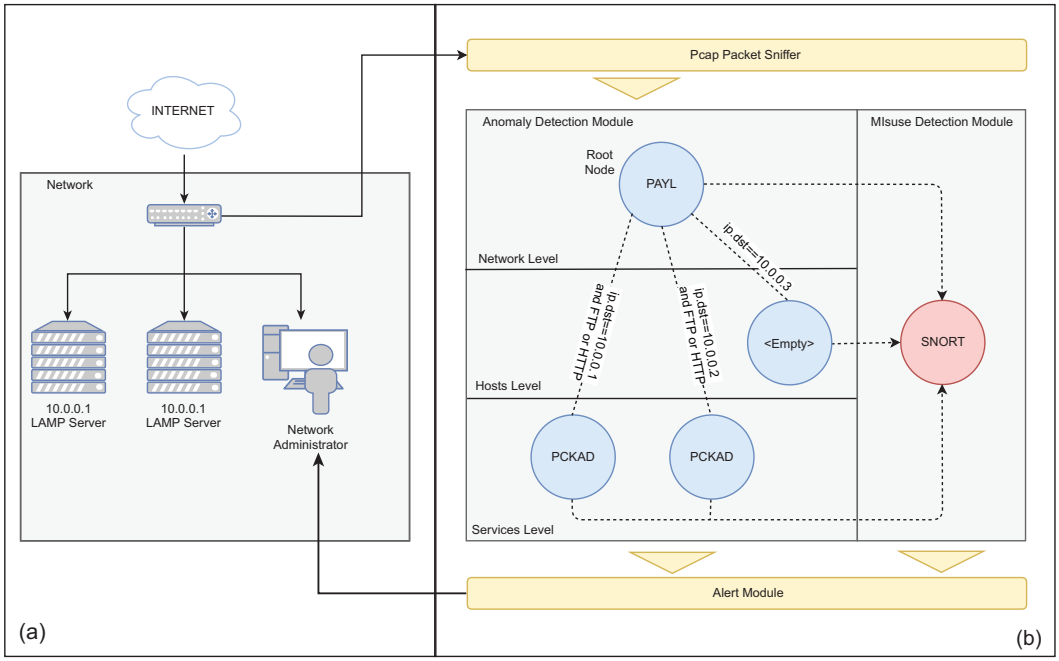


FIGURE 3.10: Configuration Scheme as Network Intrusion Detection System

- starts the SNORT process as daemon and puts it to listen on the network interface just created in non-promiscuous mode;
- opens a Unix-Socket (i.e. `/var/log/snort/snort_alert`) which it listens to obtain the analysis results.

Finally, we added a final rule in the SNORT configuration, according to which if a packet is not detected as an attack then it is labelled as normal. When a packet is delivered to the SNORT node for the analysis, it is sent on the virtual network interface and through the sockets the result is subsequently returned in output.

Before explaining how the three techniques were deployed in the framework, the network configuration used for the case study is discussed.

Figure 3.10, block *a*, shows the network topology examined. It is composed of an access router that connects three nodes to the Internet. Two of these nodes are lamp servers that offer various types of services, while the third node is control station, through which the network administrator manages the servers and network configuration. A fourth node was introduced into the network, which hosts the verticalization of the framework and analyses incoming traffic to the network. The latter node is not shown in the network topology, but it is exploded in block *b* of the figure. Block b shows how the framework was set up to protect the network. A three-level hierarchical structure was created inside the anomaly module. The first level hosts the root node, which should host a technique that can analyse all incoming traffic. It follows the hosts level, where each node is configured to be able to analyse the traffic addressed to a specific network host. Finally, each node of the last level analyses the traffic of a specific service. Since PAYL supports a greater number of protocol it was deployed in the root node. Instead, PCkAD is specific to the HTTP and FTP traffic, on which it offers the best results, therefore it was deployed in the third-level nodes as depicted in Figure 3.10. With this configuration, PCkAD can monitor the traffic routed to both network servers. The second level contains only one node, corresponding to the machine used by the network administrator. The

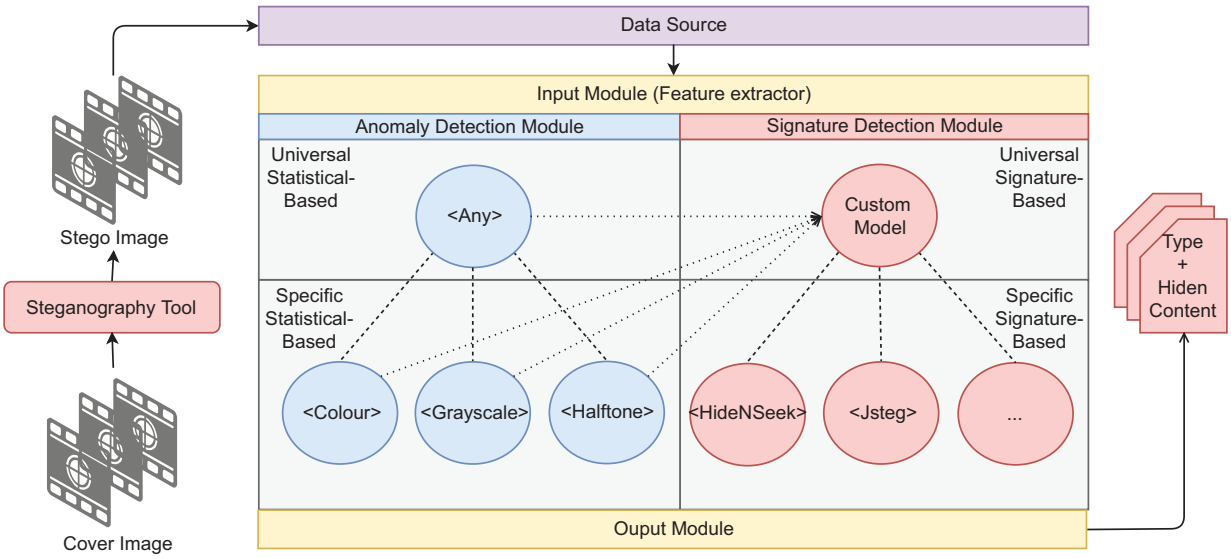


FIGURE 3.11: Configuration Scheme as Steganalysis tool

node is not associated with any anomaly detection technique because it does not host any services, and there is no need to model the administrator behaviour. All nodes of the hierarchical structure are connected to the module signature-based, which contains a single node where the Snort plugin is installed.

We developed an output module for sending reports to a network administrator via email. An email is sent when an attack is notified by a detection module, or if an host exceeds a certain predetermined threshold of anomaly.

With this case study we have easily developed a system that can detect anomalies in network traffic of separate hosts and send e-mails to a network administrator to notify the detected attacks.

3.4.2 Steganography

In this case study we configure the framework as a hybrid steganalysis tool able to find hidden content and report suspicious images.

Steganography is a technique whose objective is to hide communication between two parties, by concealing specific messages, in the form of files or text, inside of other files, such as images and video [121]. Steganography is used for example by terrorists, criminal organizations, etc., to camouflage the planning and coordination of their illicit activities.

Considering the importance of the detection of hidden messages within suspicious files, the scientific community has devoted much attention to the development of detection techniques. Two main classes of techniques have been developed, i.e. signature and statistical based. Signatures are established through the knowledge of the processes and techniques used by stenography tools for hiding content inside files. These approaches, however, are not able to detect the presence of hidden content if they are applied through ad-hoc techniques. This problem is handled through the use of statistical-based techniques that use statistics and frequencies to notice which files have anomalies. Techniques belonging to both classes can be also divided in two groups: specific and universal approaches [162]. The former targets a specific steganographic method while the latter can potentially detect all steganographic techniques.

Figure 3.11 shows how *HALF* is verticalized for the proposed case. For our purpose we developed an input module that builds an internal representation of an input image. This representation consists of *i)* a header, in which information such as image format, type of colors, and so on are specified; and *ii)* a body containing the raw image data. The header is specified to the rules engine to generate rules. The anomaly-based module is composed of two layers. The first layer contains only the root node, in which a technique for the analysis of any type of image is deployed. The second level hosts techniques each of which works on particular types of images. Given the need to analyse any type of image, regardless of the steganography technique employed, the root node hosts an universal technique. To this end, any of the techniques discussed in [162] may be suitable, such as [218] and [98]. The second layer contains a variable number of nodes, depending on how many types of image the user wants to analyse. In this specific case study it is assumed that the user is interested in halftone, grayscale and colour images. For the above analysis the techniques presented in the [117], [74] and [144] are chose, respectively.

As discussed in [118] and [162], by manually analyzing stego images, an analyst can obtain signatures that characterize the steganographic technique employed to hide a message. Then the signatures are used by specific models to conduct automatic analysis of new samples. The signature-based module has been developed according to this logic. It consists of two layers. The second layer contains nodes whose aim is to check whether a specific technique of steganography was used within an input image. Therefore, each node should host a model built for using signatures of a specific steganography technique. Once again the first layer contains only the root node. Unlike the previously discussed root nodes, this one does not host a specific technique, but a routing module. The aim of this module is to route a new anomaly image to the second-layer nodes that are able to process it, on the basis of the defined rules. Once the image gets processed, the nodes notify the result, e.g. the stego message and information about the steganography technique used, to the output module.

3.5 Summary

Nowadays the analysis of data and seeking patterns is a growing issue. In particular, this problem is very important because anomalies usually represent significant or critical information. This problem is addressed in many application domains such as medical, energy consumption and cyber security fields.

Researchers are constantly studying and developing new algorithms in order to analyse the ever increasing amount of data available and keep up with their evolution. Sometimes there might be the need to combine different types of existing techniques to achieve better results, leading to the creation of hybrid systems. This idea is similar to the concept of defence-in-depth, where multiple security tools with complementary capabilities are combined to improve the security of a target system, given that a single security tool cannot provide an holistic protection by itself. Likewise, a single anomaly detection technique has limited capabilities, in that it can typically detect one or a few classes of anomalies. That said, state-of-the-art hybrid systems are typically bound to specific techniques and limit the analysis to specific types of data. These constraints may limit the effectiveness of the analysis.

The existence of a software able to provide a basis for the analysis of data that supports the elements discussed above would make easier and more effective the approach to the problem of the anomaly detection.

Our research towards this goal led us to the design of *HALF*, a generic framework through which it is possible to combine and manage different techniques of a specific domain by means of a hierarchical structure. Its generality makes it easily applicable in different application domains such as fraud detection, speech recognition and intrusion detection for cyber security. The framework fully supports the entire data analysis process, from the training and data collection phases to the execution of data analysis algorithms. Moreover, we introduced a learning module to handle the concept drift.

In the next chapter, we present PCkAD, an intrusion detection technique. PCkAD is part of a research activity for which the concept of anomaly is of utmost importance, indeed an intrusion can be seen as an anomalous activity that is interesting to detect because it might represent a cyber threat.

Chapter 4

PCkAD: Packet Chunk Anomaly Detector

Content-based attacks is a class of attacks whose goal is to deliver malicious content (e.g. shellcode or a malicious FTP command-value pair) to a service in order to exploit its vulnerabilities. This type of cyber threats has gained considerable importance over these years, especially when their target are web applications. In this case we talk about application level web attacks. Such attacks exploit web-application vulnerabilities, to achieve goals such as data theft and privilege escalation. If the attacker succeeds, the target organization may suffer severe financial loss and damages to its image. A recent example of web attacks is the `shellshock` attack [157] which could expose organizations and individuals to potential fraud, financial loss, or access to confidential information, by exploiting a `bash` vulnerability of UNIX-like OSs.

A class of security tools that have proven to be a valuable defence wall against this type of attacks is called Intrusion Detection System or IDS. There exist two main classes of IDS [21]: *misuse-based* [180] and *anomaly-based* IDS. The former uses a database of rules or signatures to search for specific patterns in network packets to detect well-known attacks, but it fails with unknown malicious attempts or even variants of known ones. In contrast, the latter class is able to identify both known and unknown attacks, at the expense of a higher level of false positives, by modelling legitimate network traffic. This is the most known drawback of *anomaly-based* IDSs, which can lead to an excessive workload to recognize true attacks among the false positives and lose confidence in alerts.

In this chapter we present *Packet Chunk Anomaly Detector* or *PCkAD*. PCkAD is an anomaly-based intrusion detection technique thought to monitor network traffic. Its main goal is to *detect content-based attacks* by monitoring application-level traffic, e.g. web applications.

PCkAD uses the $n-$gram technique to model normal network traffic. The $n-$grams have proven to be an effective tool for the detection of malicious contents, however they alone might be insufficient to create a reliable intrusion detection system. Malicious contents tend to contain never seen $n-$grams that increase the degree of anomaly, but, on the other hand, they might also contain legitimate $n-$grams that could make them appear as normal contents.

Another very important aspect of an IDS concerns the way it has been designed. Typically, an IDS is designed with only the problem to solve in mind, i.e. detecting specific types of attack. The problem with this attitude is that nowadays in order to deploy an IDS in an organization network, it must be able to cope with the presence of an adversary attacker, i.e. a malicious user who aims at hindering the normal activities performed by the security tool. Such attackers might employ the so-called

exploratory evasion techniques [57][30] whose objective is to carefully manipulate attack samples in such a way that they seem legitimate from the IDS point of view. Section 2.3 provides an in-depth characterisation of adversarial attackers and the techniques they employ. Unfortunately, many works in literature, like [182], mainly focus on how to detect attacks and do not dedicate significant efforts to address the presence of an adversary, sometimes this aspect is not taken into account at all [34]. It is then of utmost importance to design the system in such a way that it is robust to adversary attacks.

The contributions of the approach are summarized next:

- As a major feature, differently from previous techniques that aim mainly at learning the count distribution of raw content, the proposed technique builds models of legitimate traffic by learning the spatial distribution of the n-grams occurring in the relevant payloads of normal traffic. Moreover, separate models are associated with different traffic lengths in order to better capture the relationship between length and content inherent in traffic.

- The strategy adopted has two main advantages: i) it allows the system to recognize both never seen and suspicious legitimate patterns (i.e. used in an anomalous way), and ii) it provides resistance to a specific evasion technique, known as blending. This evasion technique, which consists in obfuscating malicious code by mixing it with legitimate content, is difficult to implement by an attacker in the case of PCkAD, since both occurrence and position distributions have to be determined.

- We further prove that, even if the above information are available to the attacker, finding the right legitimate content in presence of chunks is per se a difficult problem, in that it turns out to be NP-hard.

- PCkAD is evaluated on both real and synthetic datasets. We show that PCkAD is able to achieve an high detection rate with a very low false positive rate. Moreover, we present a comparison between PCkAD and two well-known anomaly-based IDSs that use n-grams as well.

  Experimental results show that PCkAD outperforms the other two algorithms.

The source code of PCkAD is publicly available on GitHub [12].

This chapter is structured as follows. Section 4.1 presents the related work. Section 4.2 gives an overview of the technique and of its features. Section 4.3 explains the type of attacks that can be detected by PCkAD. Sections 4.4 and 4.5 explain how PCkAD models the network traffic and performs packet classification, respectively. Section 4.6 studies the computational complexity of blending with n-grams in presence of chunks. Section 4.7 discusses experimental results. Section 4.8 discusses the implications of processing encrypted traffic. Finally, Section 4.9 summarises the work.

4.1 Related work

An anomaly-based IDS learns normal behaviours to build a model which can discriminate between legitimate and malicious activities. The main benefit of this approach is that the IDS can identify both known and unknown attacks but, on the other hand, this feature comes at the expenses of a high generation of false positives, and this constitutes a relevant problem when performing IDS alerts log analysis.

One of the most important goals of the IDS research field is the mitigation of the false positive rate. Over the years, a lot of anomaly-based IDS have been devised [85].

Popular approaches for intrusion detection include Support Vector Machines [122] [119] [141], Decision Trees [212] [202] [35] [34], Naïve Bayes [27] [128], Ensemble Learning [216] [33] and Clustering [36], and n-grams [204, 182].

Promising results have been achieved by employing the n-gram technique for modelling network traffic. Other research fields, like information retrieval [1] and statistical natural language processing [42], also extensively employ n-gram based algorithms. Some of the IDS techniques based on n-grams are briefly discussed in the following. A general survey on IDS can be found in [60].

PAYL [204] builds a byte-frequency distribution model of network traffic payloads with 1-grams and unsupervised learning. A distinct model of normal traffic is built for each combination of destination port and length of the flow. During the detection phase, the current traffic is compared with a matching model by means of a simplified Mahalanobis distance measure, and an anomaly is raised if the distance exceeds a given threshold. PAYL was evaluated on the DARPA 1999 dataset. The IDS got good performances, despite using only 1-grams, featuring an overall detection rate of about 60% at a false positive rate less then 1%. However, as shown in [77], PAYL can be easily evaded by an attacker, if he manages to know the normal byte frequencies, since malicious payloads can be padded with bytes to match the distribution.

POSEIDON [38] performs the detection task by using PAYL as a basis, but with different preprocessing. POSEIDON uses the output of a Self Organizing Map (SOM) classifier for determining whether to create a separate model, instead of using the length of a packet payload. The SOM classifier aims at identifying similar payloads for a given destination address and port. The authors showed that this improvement produces less models and higher accuracy than PAYL.

ANAGRAM [205] uses a mixture of high-order n-grams, with $n > 1$, and can be considered as an evolution of PAYL. The use of high-order n-grams makes ANAGRAM less susceptible to mimicry attacks than PAYL, since it is harder to emulate such n-grams distribution, in padded bytes. Supervised learning is employed to model both normal and attack traffic,therefore a payload is classified based on both normal and attack models. One of the interesting features of Anagram is its ability to end model training by estimating the likelihood of observing new n-grams. Despite Anagram is effective at detecting certain types of attacks, it was shown that this approach suffers from overfitting [182].

In [182] the authors propose an anomaly-based technique, called Spectrogram, which models higher order sequences by means of a mixture of Markov chains. The technique is designed to learn a representation for both the content and the structure of script arguments strings. It has been thought to achieve a favourable trade-off between accuracy, generalization ability and speed. Spectrogram uses dynamic packet re-assembly to see what the target application sees. The IDS was able to achieve good detection and low false-positive rate on two datasets collected from two Columbia University's servers, however the speed of the method is very low due to the packet payload modelling.

Existing solutions tend to focus on producing a detection approach that features high detection rate and low false-positive rate, whereas the robustness against evasion attacks has been taken into consideration with different degrees of attention [205][182] or it has not been considered at all [38]. More generally, the problem is part of a significant area of research that concerns the design of pattern classifiers in adversarial environments. There is a great variety of works that aims at countering

evasion attacks, including [57][93][30][31]. Recently it has been shown that interactions between the classification system and the attacker can be modeled in non-zero-sum games, using game theory [43]. However, in order to satisfy the condition of uniqueness of the equilibrium, the two participants must meet specific conditions, moreover computationally demanding algorithms are required to find solutions. In [193][217] the goal is to learn secure classifiers by minimizing a modified version of the loss function which considers the worst case of data manipulation at test time. This approach suffers from a computationally intensive training phase.

Unlike previous works on IDSs, we extensively evaluated PCkAD in an adversarial environment and devised a solution against the exploratory attack called *blending* that does not have a negative impact on the performance of the technique. PCkAD was designed to search for legitimate n-grams that also appear in malicious contents so that it is not negatively affected by their presence. This is accomplished by learning structural information, a kind of information that has not been properly exploited so far.

4.2 Description of the technique

In the past a lot of IDSs were proposed to detect network attacks by examining either the IP or TCP header, or both of them [146], [185]. However, such strategy is not suitable for the detection of content-based attacks, which can be revealed by inspecting higher layer payloads, e.g. transport or application level data. Indeed, an HTTP-based attack could be recognized by analysing the URL of a request, while a FTP attack could use a suspicious command-value pair.

PCkAD performs its *analysis at the application level*, though it assumes as *processing unit the single network level packet*. This means that, even though PCkAD processes IP packets one at a time, it is anyway aware of the information regarding the higher-level protocols. For example, PCkAD knows whether a given packet is an HTTP fragment and also if it is part of a `GET` or of a `POST` request. This strategy has the advantage to account for relevant application-level information. As a consequence, PCkAD does not incur in the complexity due to the reassembly of the whole content of an application session, as done in other techniques (see, e.g., [182]).

PCkAD is designed as a pure intrusion detection technique, so its main goals consist in monitoring application traffic and in raising alerts when anomalous activities are identified. This is accomplished by distinguishing between two main phases, that are the training and the detection phase. The network traffic is modelled in a *semi-supervised* fashion, in that only legitimate data are employed to build classification models.

During the *training phase*, detailed in Section 4.4, the technique *learns the spatial distribution of the n-grams which typically occur in the relevant content of legitimate traffic*. In particular, the relevant content is split up in portions of equal length, called *chunks*, and the spatial distribution of patterns characterizing legitimate traffic is modelled as the expected number of occurrences of the n-grams within each chunk. Distributions are computed on groups of packets homogeneous for application-layer protocol, direction, type and, moreover, number of chunks. The last feature is inspired by studies carried out in the literature concerning the relationship between the packet length and the traffic content, as the one reported in [204] which has highlighted how the byte distribution of HTTP packets varies among different payload length.

During the *detection phase*, detailed in Section 4.5, the models built during the training phase are used to identify anomalous network traffic. Specifically, for each packet an *anomaly score* is computed and exploited to classify the packet as anomalous or not. With this aim three different detection schemes are available, which take into account global information, local information or both. This strategy allows the system to recognize both *never seen patterns* and *legitimate patterns which are used in an anomalous way*. Moreover, this feature has the additional advantage of providing *resistance to an evasion technique called blending* [77]. To illustrate, an anomaly-based IDS, which performs deep packet inspection, is able to recognize zero day attacks which contain, typically, never seen contents. When using the n-gram technique, such contents are interpreted as a set of unusual (e.g., never seen) n-grams.

It is important to note that malicious packets may contain data whose nature is not malicious, e.g. very frequent n-grams. The presence of such data could prevent an IDS to correctly detect threats by lowering the overall packet anomaly score. Thus, looking only at anomalous n-grams might be not enough. However, usually they are not distributed in the packet in the same way as in legitimate traffic. For instance, a malicious packet could contain a set of legitimate n-grams that appear with an anomalous number of occurrences and/or in unusual positions within the payload. Hence, accounting for spatial distribution may well mitigate the negative influence of legitimate data contained in malicious packets and, in addition, hinder an evasion attack such as *blending*.

When an attacker performs blending, he tries to make an attack packet appear as normal, typically by mixing malicious code with a set of legitimate bytes. The outcome of such process is an attack packet which might contain a considerable amount of legitimate content. However, *deceiving PCkAD is not an easy task* because the attacker not only has to guess the exact distribution of the n-grams, in terms of occurrences, but he also has to guess in which chunks such patterns are typically located. Moreover, we show (see Section 4.6) that even if the above information is available, *finding a legitimate content suitable to obfuscate the malicious one is per se a difficult problem in the presence of length constraints*. The behaviour of the technique with blended traffic is analysed in-depth in Section 4.7.

Table 4.1 reports a complete list of the symbols used throughout the chapter.

4.3 Content-based attacks

Content-based attacks were introduced in Section 2.1.2, while this section explains their characteristics and what they look like. One of the reasons behind the success of this class of attacks is the lack of input validation on the server.

Figure 4.1 depicts an example of a legitimate URL that is part of an HTTP GET payload (a). The URL points to *www.dangerous.com* and is followed by two parameters, namely *param1* and *param2*, both of which takes integer values. Below the legitimate payload a few examples of content-based attacks are reported. As can be seen, the difference between the malicious and legitimate payloads lies in the input

(a) http://www.dangerous.com/retrieve.php?param1=302&param2=2002
(b) http://www.dangerous.com/retrieve.asp?param1=1'%20or%20'1'%20=%20'1&param2=2002
(c) http://www.dangerous.com/retrieve.php?param1=../../../../etc/passwd&param2=2002
(d) http://www.dangerous.com/retrieve.php?param1=$include($bbb)$exit()&bbb=http://www.hxx.org/malicious.txt?&param2=2002

FIGURE 4.1: Example of legitimate HTTP GET payload (a) and content-based attacks (b,c,d)

TABLE 4.1: Symbols used throughout the chapter.

n	the n-gram length
g	n-gram
Σ^n	n-gram alphabet
l_{ck}	the chunk length
l_{rc}	the relevant payload length
n_{ck}	the number of chunks
$\mathcal{P}$	traffic profile
$\mathcal{P}_a$	application-layer protocol of $\mathcal{P}$
$\mathcal{P}_d$	packet direction of $\mathcal{P}$
$\mathcal{P}_t$	payload type of $\mathcal{P}$
$\mathcal{P}_l$	relevant payload length in terms of number of chunks of $\mathcal{P}$
T	set of packets
$T_{\mathcal{P}}$	subset of packets of T that comply with the profile $\mathcal{P}$
$M_{\mathcal{P}}$	the model associated to a profile $\mathcal{P}$
$\Sigma_{\mathcal{P}}$	the set of n-grams occurring in at least a packet in $T_{\mathcal{P}}$
$T^g_{\mathcal{P}}$	the set of packets in which g occurs at least once
$\mu^g_{\mathcal{P}}$	the mean regarding the number of occurrences of g in the packets where it appears
$\sigma^g_{\mathcal{P}}$	the standard deviation regarding the number of occurrences of g in the packets where it appears
$\mu^g_{\mathcal{P},i}$	the mean regarding the number of occurrences of g in the i-th chunk of $\mathcal{P}$
$\sigma^g_{\mathcal{P},i}$	the standard deviation regarding the number of occurrences of g in the i-th chunk of $\mathcal{P}$
p	a packet

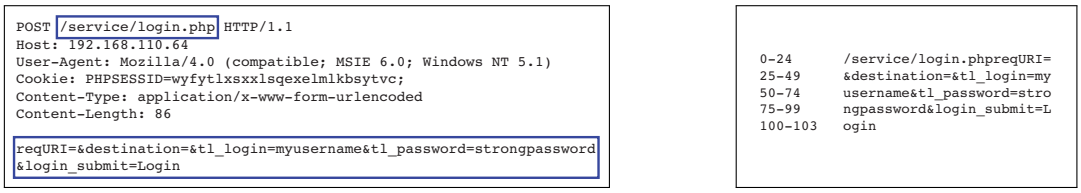
c^g_p	the number of occurrences of g in p
$c^g_{p,i}$	the number of occurrences of g in the i-th chunk of a packet p
$G_{\mathcal{P}}$	the global component of $M_{\mathcal{P}}$
$L_{\mathcal{P}}$	the local component of $M_{\mathcal{P}}$
m	$\mathcal{P}_l$
l^p_{pkt}	the size of a packet p
l^p_{ng}	the number of n-gram in p
o	a packet to classify
$\mathcal{P}^o$	the profile to which o belongs to
$\mathcal{P}^o_a$	application-layer protocol of o
$\mathcal{P}^o_d$	packet direction of o
$\mathcal{P}^o_t$	payload type of o
$\mathcal{P}^o_l$	relevant payload length in terms of number of chunks of o
$\tilde{\mathcal{P}}^o$	the most similar profile of $\mathcal{P}^o$ with respect to $\mathcal{P}^o_l$
$Z(g,o)$	the global Zeta score of g in o
$Z_i(g,o)$	the local Zeta score of g in the i-th chunk of o
$as_G(g,o)$	the global anomaly score of g in o
$as_L(g,o)$	the local anomaly score of g in o
$\mathbb{1}$	the indicator function
$as_{GS}(o)$	the anomaly score of o based on global information
$as_{LS}(o)$	the anomaly score of o based on the n-gram distribution within the packet
$as_{2LS}(o)$	the anomaly score of o based on the combination of $as_{GS}(o)$ and $as_{LS}(o)$
t_S	the n-gram anomaly score threshold
t_{as}	the packet anomaly score threshold

provided for the URL parameters, which in the former case consists of a malicious sequence of characters, like an SQL code (b). The server expects to read integer values when processing the request, but when it receives a payload like (b, c, d) it reads alphanumeric strings that could cause serious damages if not properly handled. In particular, (b) is an SQL-injection that attempts to print one or more values related to *param1*, by sending a query that is always true, i.e. $1'or'1' =' 1$. The payload in (c) is a path traversal attack whose goal is to steal a file called *passwd* from the server. The remaining attack (d) is known as *remote file inclusion*, which in this case tries to inject PHP code into the server by exploiting the PHP's remote library inclusion feature.

URL parameters do not represent the only means by which an attacker can deliver malicious contents. Figure 4.2 shows an example of a Shellshock payload [157], where the *user-agent* field of an HTTP request is exploited to remotely execute a command.

```
GET /cgi-bin/status HTTP/1.0
user-agent: () { :; }; bin/bash –c “echo vulnerable”
```

FIGURE 4.2: Example of a Shellshock payload



(A) Relevant payload

(B) Chunks for $l_{ck} = 25$

FIGURE 4.3: Example of an HTTP POST payload

4.4 Training phase

The system does not gather statistics from IP and TCP header fields but it focuses on information relevant to the specific application-layer protocol, e.g. HTTP or FTP.

PCkAD builds normal traffic models based on n-grams. An n-gram g is a sequence of n symbols belonging to a given alphabet Σ, i.e. $g \in \Sigma^n$. n-grams are extracted from an input flow by resorting to a sliding window of length n. In this work we employ byte values as the alphabet Σ.

The training phase is done off-line on a *training set* $T = \{p_1, \ldots, p_N\}$ of packets containing legitimate traffic pertaining to a user-specified application-level protocol. The training phase receives in input two parameters: (i) the length n of the n-grams to be extracted, and (ii) the chunk size l_{ck}. During this phase, each network packet goes through three main sub-phases: (1) *Preprocessing*, (2) *Packet profile identification*, and (3) *Model building*.

The following subsections detail the three steps of the *training phase*.

4.4.1 Preprocessing phase

At the beginning, PCkAD inspects the packet headers to determine the application protocol, e.g. `HTTP`, the direction (incoming or outgoing) and the specific application-level payload type. Depending on the protocol, it is possible to identify different types of application payloads. For example, in the case of HTTP traffic, packets containing GET or POST headers can be observed.

PCkAD does not process an entire application flow but single IP packets. Then some packets may contain the header of the application-level protocol while others are fragments of the payload. For these reasons, for each type of IP packet two different models are respectively built, that are *header* models and *fragment* models.

Not always the whole payload is relevant for the detection of a malicious behaviour. For example, HTTP attacks typically exploit only a portion of the entire payload of a request to carry malicious code to a vulnerable service. Such a portion is referred to as *relevant content*, and will be indicated as rc. Fig. 4.3a shows an example of payload that corresponds to an HTTP POST request header. It is composed of the start line, containing the verb POST, a set of header fields (e.g. User-Agent and Host) and the body data (see [110] for more details). Of all these elements, an attacker typically exploits the body data and sometimes the URL following the verb (e.g. [188]).

As a default setting, PCkAD considers the body data for POST requests as relevant content (that are the characters surrounded by blue rectangles in the figure),

and also the URL parameters for GET requests. However, as anticipated in Section 4.3, it might happen that other fields are exploited by an attacker, usually by forcing them to have arbitrary values. Examples of such attacks are *cross site scripting* (XSS) and *CSRF attacks* (see [214] and [125]). To capture also these attacks, PCkAD can be configured to choose which part of a packet has to be taken into account for the training and detection phase.

As for the FTP traffic, the entire payload is considered relevant by PCkAD.

The extraction of the relevant content rc carries a variety of benefits. First, the data which has to be processed by the subsequent phases is typically smaller than the whole payload and less computational resources are needed. Second, removing byte sequences having no discriminating power results in a higher classification accuracy.

After the relevant payload has been identified, PCkAD splits it up into not overlapping portions of equal length, which we refer to as *chunks*. Let l_{rc} denote the relevant payload length, the number of chunks n_{ck} that rc contains is given by: $n_{ck} = \lceil l_{rc}/l_{ck} \rceil$. Fig. 4.3b shows a possible subdivision of the relevant content of Fig. 4.3a for $l_{ck} = 25$.

4.4.2 Packet Profile Identification phase

For each packet, PCkAD identifies the matching profile and then uses the extracted data to incrementally build the corresponding model. Each profile $\mathcal{P}$ is a tuple $(\mathcal{P}_a, \mathcal{P}_d, \mathcal{P}_t, \mathcal{P}_l)$ characterized by the following attributes: *application-layer protocol* ($\mathcal{P}_a$); *packet direction*, that is either *incoming* or *outgoing* ($\mathcal{P}_d$); *payload type* ($\mathcal{P}_t$); relevant payload length in terms of *number of chunks* into which it is split up ($\mathcal{P}_l$). For example, for a chunk length of 25 bytes, the profile of the packet depicted in Fig. 4.3a is characterized by `HTTP` as the application protocol, *incoming* as the packet direction, `POST` as application specific class and 5 as number of chunks (computed on the basis of the relevant content which, in this case, is 105 bytes).

The choice of the attributes employed to characterize a profile is actually an heuristic thought to distinguish different types of traffic, similarly to what was proposed in [204].

$\mathcal{P}_a$ and $\mathcal{P}_d$ are needed in order to separate network traffic profiles of different nature, since each protocol implies specific contents and models learned on heterogeneous traffics usually exhibit large misclassification rates. We note that these two attributes alone could not be enough to build accurate models. Attributes $\mathcal{P}_t$ and $\mathcal{P}_l$ serve the purpose of better discriminating real application traffic profiles.

For example, HTTP GET and POST requests typically carry different types of data and, hence, will be grouped in different profiles. In particular, the protocol domain knowledge, taken into account by $\mathcal{P}_t$, is complemented by the packet size $\mathcal{P}_l$ which instead does not depend on the specific protocol.

Given a profile $\mathcal{P}$, we denote by $T_{\mathcal{P}}$ the subset of packets of the training set T matching with the profile $\mathcal{P}$.

4.4.3 Model building phase

During the model building phase, PCkAD builds a model $M_{\mathcal{P}}$ for each profile $\mathcal{P}$ such that $T_{\mathcal{P}} \neq \emptyset$. Given a profile $\mathcal{P}$, let $\Sigma_{\mathcal{P}}$ be the set of n-grams occurring in at least a packet in $T_{\mathcal{P}}$. Given a n-gram $g \in \Sigma_{\mathcal{P}}$, let $T^g_{\mathcal{P}} \subseteq T_{\mathcal{P}}$ be set of packets in which g occurs at least once.

For each $g \in \Sigma_{\mathcal{P}}$, PCkAD computes the mean and the standard deviation of its occurrences among the packets where it appears:

$$\mu_{\mathcal{P}}^{g} = \frac{1}{|T_{\mathcal{P}}^{g}|} \sum_{p \in T_{\mathcal{P}}^{g}} c_{p}^{g} \text{ and } \sigma_{\mathcal{P}}^{g} = \left(\frac{1}{|T_{\mathcal{P}}^{g}|} \sum_{p \in T_{\mathcal{P}}^{g}} (c_{p}^{g} - \mu_{\mathcal{P}}^{g})^{2} \right)^{\frac{1}{2}}, \tag{4.1}$$

and the mean and the standard deviation of the number of occurrences for each chunk:

$$\mu_{\mathcal{P},i}^{g} = \frac{1}{|T_{\mathcal{P},i}^{g}|} \sum_{p \in T_{\mathcal{P},i}^{g}} c_{p,i}^{g} \text{ and } \sigma_{\mathcal{P},i}^{g} = \left(\frac{1}{|T_{\mathcal{P},i}^{g}|} \sum_{p \in T_{\mathcal{P},i}^{g}} (c_{p,i}^{g} - \mu_{\mathcal{P},i}^{g})^{2} \right)^{\frac{1}{2}}, \tag{4.2}$$

where c_p^g is the number of occurrences of g in a packet $p \in T_{\mathcal{P}}^{g}$, $c_{p,i}^{g}$ represents the number of occurrences of g in the i-th chunk of a packet p and $T_{\mathcal{P},i}^{g}$ is the set of packets where g occurs in the i-th chunk.

A model $M_{\mathcal{P}}$ of a profile $\mathcal{P}$ is a pair $M_{\mathcal{P}} = (G_{\mathcal{P}}, L_{\mathcal{P}})$, where $G_{\mathcal{P}}$ is the global component and $L_{\mathcal{P}}$ the local component.

The global component $G_{\mathcal{P}}$ of a model $M_{\mathcal{P}}$ is a set of pairs $G_{\mathcal{P}} = \{(\mu_{\mathcal{P}}^{g}, \sigma_{\mathcal{P}}^{g}) \mid g \in \Sigma_{\mathcal{P}}\}$, each one associated with a different n-gram g occurring in $\Sigma_{\mathcal{P}}$.

The local component $L_{\mathcal{P}}$ of a model $M_{\mathcal{P}}$ consists of a set of m-tuples $L_M = \{\langle(\mu_{\mathcal{P},1}^{g}, \sigma_{\mathcal{P},1}^{g}), \ldots, (\mu_{\mathcal{P},m}^{g}, \sigma_{\mathcal{P},m}^{g})\rangle \mid g \in \Sigma_{\mathcal{P}}\}$, each one associated with a different n-gram g occurring in M, where $m = \mathcal{P}_l$.

4.4.4 Temporal and spatial cost

The dimension of the input $|I|$ corresponds to the size of the training set T, that is $|I| = \sum_{p \in T} l_{pkt}^{p}$, where l_{pkt}^{p} is the byte length of packet p.

The preprocessing step is done in linear time with respect to l_{pkt}^{p}, while the profile identification step requires constant time. During the model building step, the model building operation accesses the data structure which stores model information, so we must account for that access time. We employ a hash table for storing models and, as a consequence, we achieve $O(1)$ as the amortized cost for accessing it. Then, because the hash table has to be accessed for each n-gram extracted from the relevant content of p, the cost of the third step is $O(l_{ng}^{p})$, where $l_{ng}^{p} = l_{rc}^{p} - n + 1$, with l_{rc}^{p} the size of the relevant content of p, i.e. the total number of n-grams contained in the relevant content of p. Being $l_{ng}^{p} < l_{pkt}^{p}$, the time complexity of the training phase is $O(|I|)$, that is to say linear in the number of bytes of the training set.

Regarding the spatial complexity, the data structure where models are stored dominates by far the space usage. In the worst case, each chunk requires a distinct model and then $O(|I|/l_{ck})$ space is need. However, we note that in practice the space requirements are much smaller.

4.5 Detection phase

PCkAD exploits models $M_{\mathcal{P}}$ for classifying new observations with the goal of detecting content-based attacks. Packet classification is also done in three steps: (1) packet preprocessing, (2) profile identification and (3) anomaly computation. The first two are the same as in the training phase. The third one is detailed in the following.

4.5.1 Anomaly scores and detection strategies

Each observed packet o is first parsed in order to identify the presence of content not structured accordingly to what is specified by the application protocol. For example, when PCkAD inspects a payload containing an HTTP GET header, it is aware of how the packet must be structured (the format of HTTP requests is described in the protocol specification `RFC 2616` [110]).

If the packet o is not well-structured, it is automatically marked as anomalous, otherwise it is compared with the model $M_{\mathcal{P}^o}$ corresponding to the matching profile $\mathcal{P}^o = (\mathcal{P}^o_a, \mathcal{P}^o_d, \mathcal{P}^o_t, \mathcal{P}^o_l)$. In the case there is no matching profile but there are some differences only for the $\mathcal{P}_l$ component, it is still possible to try to classify o by using the model $M_{\tilde{\mathcal{P}}^o}$ for the profile $\tilde{\mathcal{P}}^o = \arg\min_{\mathcal{P}}\{|\mathcal{P}_l - \mathcal{P}^o_l| : (\mathcal{P}_a = \mathcal{P}^o_a) \wedge (\mathcal{P}_d = \mathcal{P}^o_d) \wedge (\mathcal{P}_t = \mathcal{P}^o_t)\}$ characterized by the most similar number of chunks.

We exploit the trained models in order to compute the Zeta score, which measures how many standard deviations the observed value is far apart from the population mean. Given an observation o, the *global* Zeta score $Z(g, o)$ of the n-gram g in o is

$$Z(g, o) = \left| \frac{c^g_o - \mu^g_{\mathcal{P}^o}}{\sigma^g_{\mathcal{P}^o}} \right|. \tag{4.3}$$

Analogously, a set of of *local* Zeta scores $Z_i(g, o)$ $(1 \leq i \leq \mathcal{P}^o_l)$ are associated with g in o by considering each different chunk within the packet:

$$Z_i(g, o) = \left| \frac{c^g_{o,i} - \mu^g_{\mathcal{P}^o,i}}{\sigma^g_{\mathcal{P}^o,i}} \right|. \tag{4.4}$$

Now, we introduce the anomaly scores we employ to detect anomalous traffic. Anomaly scores make use of a threshold t_S. The global anomaly score of g in o is defined as:

$$as_G(g, o) = \begin{cases} c^g_o & \text{if} \quad Z(g, o) > t_S \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The local anomaly score of g in o is defined as:

$$as_L(g, o) = \sum_i \Big(c^g_{o,i} \cdot \mathbb{1}[Z_i(g, o) > t_S] \Big), \tag{4.6}$$

where $\mathbb{1} : \{f, t\} \mapsto \{0, 1\}$ denotes the indicator function s.t. $\mathbb{1}(f) = 0$ and $\mathbb{1}(t) = 1$. In those cases where there is no model associated with $\mathcal{P}^o$, we use the model $M_{\tilde{\mathcal{P}}^o}$ and the local anomaly score becomes:

$$as_L(g, o) = \sum_{i=1}^{\tilde{\mathcal{P}}^o_l} \Big(c^g_{o,i} \cdot \mathbb{1}[Z_i(g, o) > t_S] \Big) + \sum_{i=\tilde{\mathcal{P}}^o_l+1}^{\mathcal{P}^o_l} c^g_{o,i}, \tag{4.7}$$

that is to say, we consider as anomalous the n-grams occurrences pertaining to chunks not shared with the model.

We devised three different strategies for deeming an observed packet as anomalous. l^o_{ng} denotes the total number of n-grams contained in the relevant content of o.

Global strategy (GS). This strategy takes into account global information:

$$as_{GS}(o) = \frac{1}{l^o_{ng}} \sum_g as_G(g, o). \tag{4.8}$$

It is the fastest strategy and has the lowest memory requirements.

Local strategy (LS). It takes into account the n-gram distribution within the packet:

$$as_{LS}(o) = \frac{1}{l^o_{ng}} \sum_g as_L(g, o). \tag{4.9}$$

It has the benefit of taking into account a greater amount of structural information than GS and of enforcing stricter constraints on the payload structure. Hence, We expect such a strategy to be more accurate than GS.

Two levels strategy (2LS). This strategy combines the two above defined strategies:

$$as_{2LS}(o) = \frac{1}{l^o_{ng}} \sum_g \big(as_G(g, o) + as_L(g, o) \cdot \mathbb{1}[Z(g, o) \leq t_S]\big). \tag{4.10}$$

It is a hybrid strategy which can be faster than LS and more accurate than GS.

A packet o is declared as anomalous if its anomaly score exceeds a give threshold t_{as}.

As for the temporal cost of the detection phase, all the anomaly scores can be computed in time $O(l^o_{ng})$, i.e. linear in the length of the relevant content of the packet.

4.6 The complexity of blending with chunks

Blending a packet requires to figure out how to pad malicious payloads with fake data so as to obtain a n-gram distribution which is similar to that of a legitimate payload. This is in general not easy because the padding must not alter the malicious payload semantics while guessing the exact distribution of the n-grams employed to learn the classification models.

Moreover, here we show that even if all the information exploited by the IDS to detect intrusions is known to the attacker, the blending problem in presence of chunks still remains intrinsically difficult. More formally, given a malicious payload s_{mal} the attacker is interested to position in a piece of traffic (chunk), of given length K, a padding byte sequence s_{pad}, of a proper length $K' = K - |s_{mal}|$. The sequence s_{pad} should be crafted so as to minimize the anomaly score of the blended content $s_{mal} \oplus s_{pad}$, where $\oplus$ denotes the blending operator, that is

$$s^*_{pad} = \arg \min_{s_{pad} \in \Sigma^{K'}} as(s_{mal} \oplus s_{pad}).$$

In order to characterize the computational complexity of the blending problem for intrusion detection techniques exploiting n-gram statistics in presence of chunks, we introduce the decision version of the above optimization problem.

Problem 4.6.1 *The* Chunk n-Gram Blending Problem $\langle s_{mal}, K, n, \Sigma, M, t\rangle$ *is defined as follows: given a malicious payload* s_{mal}*, a chunk length* $K > |s_{mal}|$*, an integer* $n > 0$*, an alphabet* Σ*, a model* M*, consisting of a certain number of* n*-grams* g_i *from the alphabet* Σ *with associated mean* μ_i *and standard deviation* σ_i *values, and a threshold* $t > 0$*, decide if*

there exists a padding payload s_{pad} of size $K-|s_{mal}|$ such that the payload $s_{mal}\ s_{pad}$ consisting of the juxtaposition of s_{mal} and s_{pad} is such that $as(s_{mal}\ s_{pad}) > t$.

We notice that the blending operator $\oplus$ is no less difficult than the juxtaposition, since the latter is a specific instance of the former. Indeed, in its general form $\oplus$ takes in input h stripes $s_{1,mal}, \ldots, s_{h,mal}$ of malicious content to be possibly interleaved with padding content.

The following results account for the intrinsic difficulty of Problem 4.6.1. Specifically, we show that this problem has connections with the Superstring problem which is routinely used in DNA sequencing and data compression practice.

Theorem 4.6.1 *The Chunk n-Gram Blending Problem is NP-hard.*

Proof. We show that the NP-complete *Superstring Problem* can be reduced to the *Chunk n-Gram Blending Problem*.

A *superstring* of a set of strings $S = \{s_1, \ldots, s_m\}$ is a string s containing each s_i, $1 < i < m$, as a substring. The *Superstring Problem* $\langle S, K\rangle$ is [83]: Given a set of strings S and a positive integer K, does S have a superstring of length K ?

We exploit the following results from [83] in order to perform the reduction.

Lemma 4.6.2 *The superstring problem is NP-complete. For two-symbol alphabets, this problem is NP-complete even if for any real number $h > 1$, all strings in S have length $\lceil h\lceil \log_2 \sum_i |s_i|\rceil\rceil$ (Th. 3 of [83]).*

Let Σ_S denote the set $s_1 \cup \ldots \cup s_m$. The malicious payload s^S_{mal} is obtained as

$$s^S_{mal} = s_1\ u\ s_2\ u\ \ldots\ u\ s_{m\text{-}1}\ u\ s_m\ u$$

and is of size $(n+1)m$, where u is a padding character not occurring in Σ_S. Since original strings s_i are different and separated by the novel padding character u, it holds that each s_i occurs exactly once in s^S_{mal}.

Let Σ^S be $\Sigma_S \cup \{u\}$. W.l.o.g. assume that $|s_i| = n$ for each string s_i in S (see Th. 1 of [83]). Moreover, let s_{pad} be a sequence of K symbols from the alphabet Σ^S.

Let g_i denote a generic n-gram on the alphabet Σ^S which does not occur in S (and possibly including u). The maximum number of occurrences of g_i in the chunk $s^S_{mal}\ s_{pad}$ is $c'_{max} = m + K - n + 1$, where $K - n + 1$ (m, resp.) is the maximum number of occurrences starting in s_{pad} (s^S_{mal}, resp.).

Moreover, assume that s_{pad} contains at least one occurrence of each s_i in S. Consider a generic string s_i of S: in order to accommodate the remaining $m-1$ n-grams, at least $m-1$ consecutive positions are needed and at most $c''_{max} = K - (m-1) + 1$ copies of s_i can be accommodated in the whole chunk (one copy is already included in s^S_{mal}.)

Given an interval $I = [a, b]$, let $\mu_I = \frac{a+b}{2}$ and $\sigma_I = \frac{b-a}{2}$. Consider the intervals $I' = [1, c'_{max}]$ and $I'' = [2, c''_{max}]$. Let the model M^S such that each $g_i \subset \left(\Sigma^S\right)^n \setminus S$ ($g_i \in S$, resp.) has mean $\mu_{I'}$ ($\mu_{I''}$, resp.') and standard deviation $\sigma_{I'}$ ($\sigma_{I''}$, resp.). This means that n-grams $g_i \subset \left(\Sigma^S\right)^n \setminus S$ do not contribute to the anomaly score, while n-grams $g_i \in S$ give a contribution > 0 to the anomaly score provided that they occur exactly once in the overall chunk or, equivalently, do not occur at all in s_{pad}.

W.l.o.g. assume that $(A1)$ $|\Sigma_S| = 2$ (and, hence, $|\Sigma^S| = 3$), that $(A2)$ each s_i is such that $|s_i| = n$ and, moreover, that $(A3)$ $m = \frac{2^n}{2n}$ or, equivalently, that $n = 1 + \log_2 mn$, where mn denotes the size (in bits) of the set S. Then the size of the input is mn. In such a case, the number of n-grams in the model M^S is $|\Sigma^S|^n = 3^{1+\log_2 mn}$, a quantity which represents a polynomial in mn, being $(mn)^2 > 3^{\log_2 mn}$ (indeed $n^2 > 3^{\log_2 n}$ if and only if $2\log_2 n > \log_2 n \cdot \log_2 3$ if and only if $2 > \log_2 3 \approx 1.585$).

Let $K^S = m(n+1)+K$. Given an instance $\langle S, K\rangle$ of the Superstring Problem satisfying assumptions $A1$+$A2$+$A3$, consider the associated instance $\langle s^S_{mal}, K^S, n, \Sigma^S, M^S, 0\rangle$ of the Chunk n-Gram Blending Problem. We note that the size of the latter instance is polynomially related to the size of the former one, since it is enough to assume $K \leq mn$, all the numbers encoded by using $O(\log_2 mn)$ bits, and all the symbols in Σ^S encoded by using 2 bits. Moreover, by Lemma 4.6.2 it holds that deciding if $\langle S, K\rangle$ is an instance YES of the Superstring Problem is NP-complete.

Now we show that $\langle S, K\rangle$ is an instance YES of the Superstring Problem if and only if $\langle s^S_{mal}, K^S, n, \Sigma^S, M^S, 0\rangle$ is an instance YES of the Chunk n-Gram Blending Problem. This will complete the proof.

($\Rightarrow$) Assume that $\langle S, K\rangle$ is an instance YES of the Superstring Problem, then there exists a string s^* of length K consisting of symbols from Σ_S which contains at least one occurrence of each s_i in S. It holds from what stated above that $s^*_{pad} = s^*$ is such that the chunk $s^S_{mal}\, s^*_{pad}$ has anomaly score > 0 with respect the model M^S.

($\Leftarrow$) Assume that $\langle s^S_{mal}, K, n, \Sigma^S, M^S, 0\rangle$ is a YES instance of the Chunk n-Gram Blending Problem, then from what stated above it is the case that there exists a sequence s^*_{pad} of K symbols from Σ^S which contains at least once each string s_i in S. It follows that the string s^* obtained from s^*_{pad} by replacing each occurrence of u with any symbol in Σ_S, is a superstring of S having size K. □

The result of Theorem 4.6.1 suggests that deceiving n-gram based techniques in presence of chunks can be far more difficult with respect to the case in which statistics simultaneously involve the whole packet content and no constraint is imposed on the size of the content pertaining to a specific model.

From the technical point of view this can be explained by the fact that chunks introduce a strict constraint that may favour the combinatorial explosion of the symbol arrangements to be explored in order to put blending into practice.

4.7 Experimental results

The section is organized as follows. The datasets are presented in Section 4.7.1. The sensitivity of the technique with respect to parameters is studied in Section 4.7.2. The effectiveness against exploratory evasion techniques is evaluated in Section 4.7.3. A comparison with two well-known intrusion detection techniques based on n-grams, i.e. Spectrogram [182] and PAYL [204], is presented in Section 4.7.4. Finally, section 4.7.5 discusses peculiarities of PCkAD strategies on the light of experimental results.

4.7.1 Dataset

In the experiments we employed the following datasets: *Tiki Usenet* (TW): It is a synthetic dataset generated by a purposely developed framework [37] by a research team of the Columbia University; *Unical Web* (UW): This dataset is made of inbound network traffic to our departmental web server at University of Calabria collected over a week; *CLET*: It is a publicly available HTTP attack dataset which was originally provided by [112] and was later enriched with new attack instances by [169]; *CSIC 2010* (CS): A publicly available HTTP dataset which was created at the "Information Security Institute" of CSIC (Spanish Research National Council) [196]. It contains web traffic concerning an e-commerce web application.

The modular framework used to produce the first dataset is called *Wind Tunnel* [37] and has been designed for synthetic data generation towards web applications. The goal of *Wind Tunnel* is to allow the comparison of security tools which operates at

different layers, e.g. a web content anomaly detector and a file access sensor. Whilst the datasets created by *Wind Tunnel* are synthetic like the DARPA 1999 dataset [142], they are much similar to real traffic because their generation is based on publicly available contents like usernames, passwords and plain English text. The authors focused only on a single important attack vector, i.e. remote attacks on web applications, to better model the content.

Of the nine available datasets created with *Wind Tunnel*, we evaluated PCkAD on *Tiki Usenet* (TW) since it is the one for which Spectrogram achieved the best performances. TW contains few HTTP attack instances of two types: (i) exploitation of a vulnerable version of the PHP function `unserialize()` [156], which allows arbitrary code execution, and (ii) an SQL injection vulnerability.

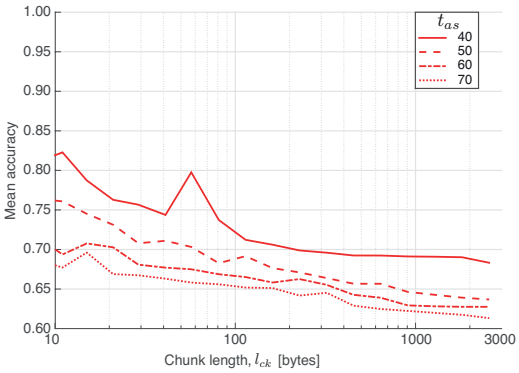
Being able to conduct experiments on a dataset made of real traffic is of utmost importance in order to get reliable results. For this reason a second dataset was built, by sniffing the network traffic from our departmental web server, for about one week. The server hosts the Department website, a number of sites reserved to academic personnel and a platform for courses management and e-learning. We selected only those packets holding `GET` or `POST` requests with parameters, hence obtaining a set of 160,000 packets. Manual labelling was performed on the dataset, with the help of domain experts that identified different kinds of attacks, such as `SQL injection` and `OS Command Injection`.

CLET is composed of three datasets each of which contains specific types of attacks: *Generic attacks* (GN): This dataset includes all the HTTP attacks provided by the authors of [112] plus a shell-code attack that exploits the vulnerability MS03-022 of the Windows Media Service (WMS); *Shell-code attacks* (SH): This dataset contains shell-code attacks; *CLET attacks* (CL): This dataset contains polymorphic attacks. It is named after the polymorphic engine CLET [64] used to generate the attacks. UW was employed to build a normal traffic dataset for CLET.

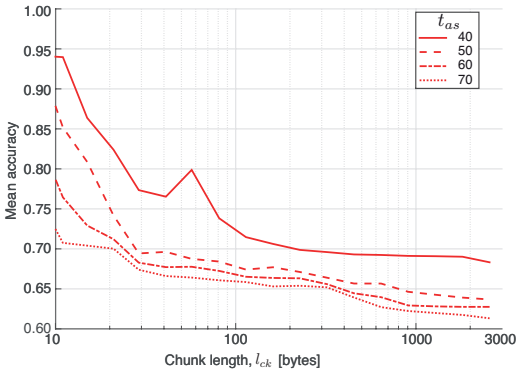
CSIC contains thousands of both normal and malicious web requests that were generated in an automatic fashion. The dataset is labelled and includes a considerable variety of attacks such as SQL injection, buffer overflow, files disclosure, CRLF injection and so on.

We also employed a purposely crafted variant of the UW dataset, called *UWB*, where malicious packets were replaced by a suitably *blended* version of them.

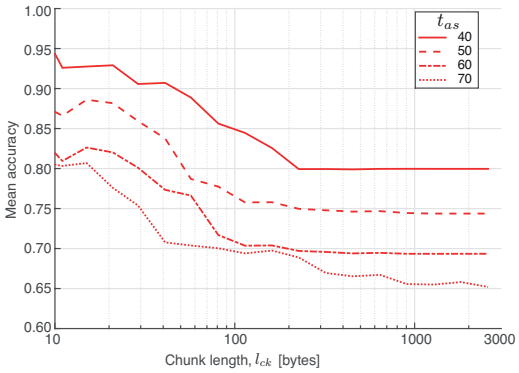
The following metrics were used for the evaluations: the detection rate (DR) (or True positive rate), i.e. the proportion of positive (anomalous) samples correctly classified, and the False positive rate (FPR), i.e. the proportion of negative samples erroneously classified as being positive. The datasets were characterised by imbalanced classes, with most of the instances belonging to the negative class or legitimate traffic. Therefore, we employed the average accuracy, i.e. the arithmetic mean of the accuracies (ratio between the correctly classified packets and the total number of packets) associated with the negative class and the positive class (or the attack traffic). Ten fold cross validation was used for all the techniques. For the experiments we used the values of n ranging in the set $\{2, 3, 4, 5\}$. Another parameter of great interest for the assessment of PCkAD is l_{ck}. By varying l_{ck} the distribution of the structural information changes, as well as the models accuracy, so we wanted to investigate how this parameter affects PCkAD from an accuracy perspective. With regard to the values of l_{ck}, a succession of logarithmically spaced values was defined, starting from a configuration which splits packets in a lot of chunks, until we reach a configuration in which the packets are made of one single chunk. If not otherwise specified, the value for the threshold t_S is 2.



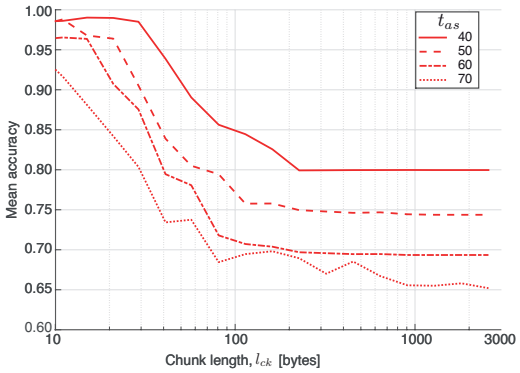
(A) GS, $n = 2$.



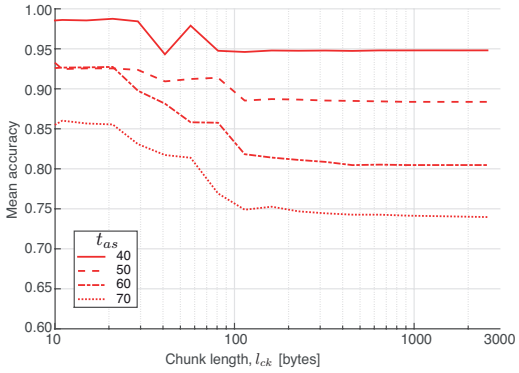
(B) LS, $n = 2$.



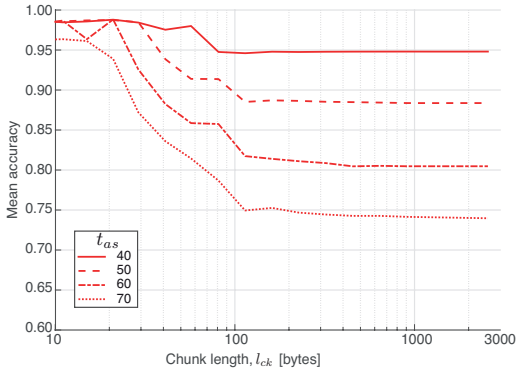
(C) GS, $n = 3$.



(D) LS, $n = 3$.

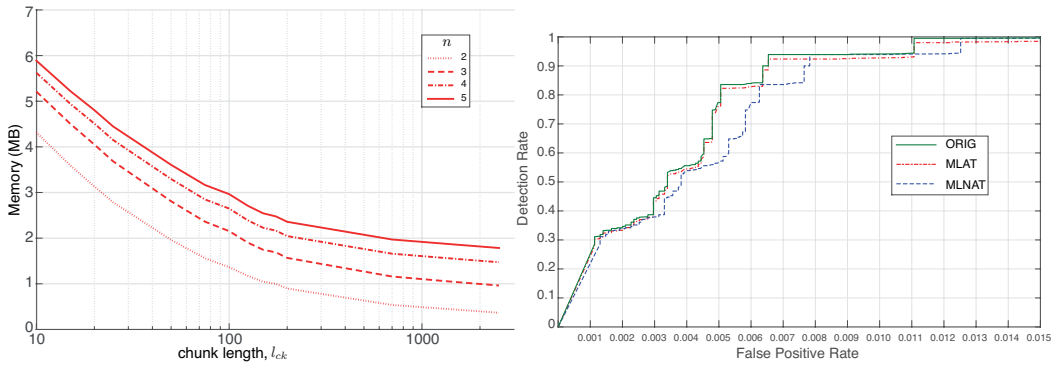


(E) GS, $n = 4$.



(F) LS, $n = 4$.

FIGURE 4.4: Dataset UW: sensitivity analysis.



(A) (B)

FIGURE 4.5: (a) Memory consumption of PCkAD on UW with respect to n-gram length and chunk length l_{ck}. (b) Robustness against meaningless traffic.

4.7.2 Sensitivity analysis

This section investigates how parameters' values affect the technique. Figure 4.4 reports the average accuracy on the UW dataset of LS (column on the right) and GS (column on the left) for $n = 2$ (upper row), $n = 3$ (middle row), and $n = 4$ (lower row), when both the chunk length l_{ck} (on the x-axis, between 10 and 3000) and the anomaly score threshold t_{as} (the four curves on each plot are associated with values $t_{as} \in \{40, 50, 60, 70\}$) vary.

We do not report the curves associated with 2LS, since in these experiments they are very close to those associated with LS.

As for the dependency from the chunk length l_{ck}, the plots highlight that the accuracy gets worse when the chunk size is comparable to the packet payload length, thus confirming that taking into account chunks is beneficial to the quality of the detection. The best trade-off between chunk length and complexity of the models is obtained for some intermediate values of size, specifically of the order of some tens of bytes.

As for dependency from the n-gram length, the shorter the n-grams, the smaller their discriminating power. Nonetheless, the longer the n-grams, the larger the peril to overfit the training data. Even for $n = 3$ performances appear to be satisfactory and sensibly better than for $n = 2$, while for $n = 5$ the behaviour is similar to the case $n = 4$. It follows that using larger values of n is not advised, due to the risk of overfitting and the increase of model complexity.

Consider the dependency from the anomaly score threshold t_{as}. Large values of t_{as} are expected to generate almost no false positives, with the accuracy of the negative class close to 1 and, hence, the average accuracy will tend to the detection rate (the accuracy of the positive class) associated with outstanding true positives. Conversely, small values of t_{as} are expected to capture larger fractions of true positives and, hence, to improve the detection rate, while the number of false positive will increase. Overall, the average accuracy tends to increase as t_{as} decreases, since the fraction of true positives increases quickly than the fraction of false positives, though for small values of t_{as} the latter fraction is expected to abruptly enlarge. As for the differences between GS and LS it appears that the latter strategy is more robust to threshold variations in the range of values for l_{ck} which guarantees the best performances.

Figure 4.5a shows the memory consumption of the LS strategy for different combinations of the n-gram length and of the chunk length l_{ck}. Given that the training

set size is about 120 MB, in the worst case, i.e. for $n = 5$ and $l_{ck} = 10$, PCkAD requires about 5.0% of the total size, while intermediate cases even halve memory requirements. 2LS uses a little more memory than LS, while GS is slightly lighter than LS.

4.7.3 Robustness against evasion attacks

The goal here is to show that the exploitation of local information makes our technique robust to the attempt of hiding attacks by reproducing the legitimate frequency of n-grams within the overall packet payload. Section 4.7.3 describes the blending technique and Section 4.7.3 reports the results of the experiments.

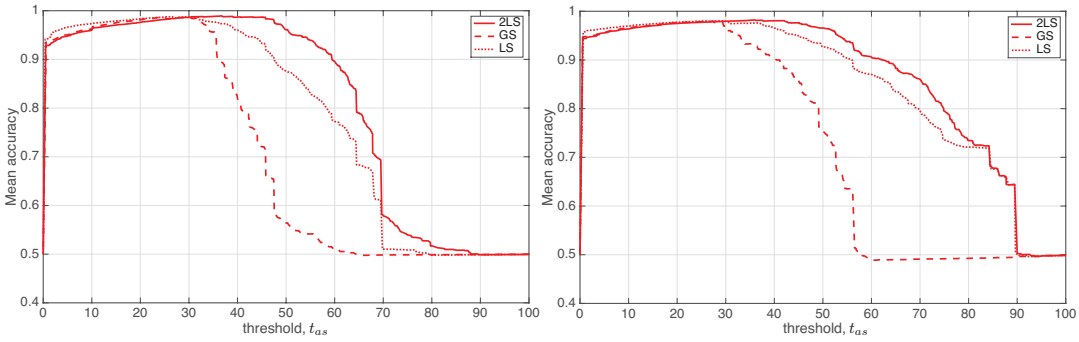
Blending technique

In [77] it was shown an approach to carry out mimicry attacks against PAYL [204] and other anomaly detectors. PAYL uses only 1-grams, so in order to bypass its inspections it was sufficient to guess how each single byte is distributed within the normal traffic. In the case of PCkAD, high order n-grams are used, therefore the simple concatenation of legitimate n-grams is not effective because it easily leads to the generation of never seen n-grams or known n-grams with an anomalous number of occurrences.

In the following we assume that the attacker (i) is able to sniff the traffic and hence to gather statistics about it, and (ii) is aware that the IDS employs a technique based on n-grams and that specific models are built for different traffic profiles. Even in this scenario, the attacker is not aware of the chunk length used by PCkAD, therefore he does not know the exact profile which the payload belongs to. Choosing the wrong profile would mean blending the malicious code with n-grams that could not be legitimate for the actual packet profile. To overcome this problem, the attacker may try to perform the blending process for a reasonable range of chunk length values l_{ck}.

Finding the optimal payload s^*_{pad} for blending the malicious payload s_{mal} is a complex task, as witnessed by Theorem 4.6.1, hence we developed an heuristic to cope with this problem. Let M denote a generic global model associated with the IDS incoming traffic. Each packet p belonging to the profile P_M associated with M is processed in order to check whether it contains a suitable s_{pad} sequence of length $K' = l^p_{pkt} - |s_{mal}|$. In particular, the blender extracts from p all possible subsequences s^p_i of length K', each one starting at position $i \in [1, l^p_{pkt} - K' + 1]$, and selects those such that the anomaly score $as_{GS}(s_{mal} \oplus s^p_i)$ is above a pre-defined threshold. From each sequences s^p_i satisfying the above condition an attacking packet is generated as $s_{mal} \oplus s^p_i$.

To illustrate the blending operator, we describe a specific type of blending concerning SQL injection. Usually, this kind of attack is characterized by the presence of strings such as "`' 1'='1`", that are exploited for extrapolating data without the right permission. An attacker could replace the 1s with two identical sequences of legitimate n-grams, so that the resulting anomaly score is lower than the selected threshold. Note that the previous example is an exceptional case, in fact it is required to duplicate the sub payload, so the blender has to take into account sub-payloads s^p_i of length $K'/2$.



(A) $n = 3$ (B) $n = 5$

FIGURE 4.6: Results of the blending experiment.

Blending results

The experimental activity on the UWB dataset, that is the version of the UW dataset blended by means of the procedure described in Section 4.7.3, allowed us to further assess the effectiveness of chunk-based strategies. Specifically, UWB contains all the legitimate traffic of UW, but the attack set was replaced by a set of about 500 blended attack packets.

Based on the sensitivity analysis presented in Section 4.4, we selected $l_{ck} = 21$ as the chunk length and varied the parameter n in the range $[3, 5]$. Moreover, we studied how the accuracy of the system changes with respect to the threshold parameter t_{as}. In the following we suppose that the attacker knows the chunk length used by PCkAD and, hence, all the blended packets are generated by the attacker by assuming $l_{ck} = 21$.

Figure 4.6 reports the impact of the blending technique on the average accuracy of the three strategies. The plots show that LS and 2LS exhibit a better accuracy than GS, with 2LS obtaining the best results for almost all the thresholds' values t_{as}. This result witnesses that the global strategy alone is not sufficient to guarantee a reliable detection for significant levels of anomaly score threshold, and that the local strategy offers sensible improvements in terms of robustness of the technique. Moreover, by merging the global and the local strategies the best performances are achieved.

Meaningless traffic injection

Here, we test the robustness of PCkAD in the case the attacker injects meaningless n-grams within the traffic for the purpose of deceiving the detection capability of the technique. With this aim we generated a novel attack dataset, called UWM, starting from the UW dataset, as follows. We randomly partitioned the legitimate traffic into ten sets and then we injected an increasing fraction, varying from the 10% to the 100% by 10%, of meaningless traffic into each set.

We considered three test sets. The first, namely ORIG, corresponds to the original UW attack set and does not include the meaningless traffic. In the second test set, namely MLAT, both the meaningless traffic and the original attacks of UW are considered as true attacks. The third one, namely MLNAT, includes the meaningless traffic, but it is considered as normal this time. Figure 4.5b reports the ROC curves associated with the three test sets. It can be seen that the accuracy of PCkAD on MLAT (AUC=0.9960) and MLNAT (AUC=0.9958) only slightly decreases with respect to ORIG (AUC=0.9964). In particular, the experiment witnesses that the

TABLE 4.2: Comparison among PCkAD, PAYL and Spectrogram based on the AUC

Dataset	UW	TW	GN	SH	CL	CS
PCkAD	99.64%	99.99%	99.91%	99.91%	99.93%	99.98%
PAYL	83.72%	80.06%	94.97%	98.53%	98.40%	86.34%
Spectrogram	86.70%	96.26%	87.68%	88.00%	99.91%	92.78%

technique performances do not significantly deteriorate in presence of meaningless traffic.

4.7.4 Comparison with PAYL and Spectrogram

In this section we compare PCkAD with PAYL [204] and Spectrogram [182]. Based on the considerations presented in [204], we chose the following values for the configuration parameters of PAYL: smoothing factor $\alpha = 0.001$ and clustering threshold $t_{CL} = 0.5$. As for Spectrogram, based on what was observed in [182], we considered the following configuration parameters: for the n-gram length the set $\{3, 5, 7, 10, 11, 13\}$ and for the number of employed Markov chains m the set $\{4, 5, 8, 10\}$. As for PCkAD, 2LS was used, with $n = 3$ and l_{ck} varied on the whole range of legal values. For each technique and associated combination of parameters, we computed the respective AUC by suitably varying the decision thresholds. The best AUCs associated with each method are shown in table 4.2.

PCkAD outperforms both PAYL and Spectrogram in every case and obtains an AUC very close to 1. Generally, PAYL performs the worst, except for the GN and SH datasets for which it achieves better performances than Spectrogram. We believe that such results are determined by the amount of structural information that a technique leverages in order to classify an observation. PAYL does not take into account structural information at all and, in addition, by relying on 1-grams, it might not be able to detect complex malicious patterns. Spectrogram, on the other hand, has a limited amount of structural information, that allows it to obtain good performances. Specifically, the technique is able to infer whether a string should follow another one by determining the normality of a character based on the characters that occur in the previous $n - 1$ positions. However, Spectrogram lacks the concept of spatial distribution, indeed the presence of legitimate n-grams within a malicious payload, regardless of their position, may lead to the computation of low anomaly scores. Finally, PCkAD is aware of the spatial distribution of the n-grams, therefore it can count on a sufficient amount of structural information that allows it to achieve the best performances.

A better AUC on TW for Spectrogram is reported in [37] . However, in that work Spectrogram is used in conjunction with a framework, called STAND [55], which has been designed to sanitize the training data in order to improve the accuracy of anomaly detection.

4.7.5 Discussion

The sensitivity analysis shows that LS and 2LS in general obtain more satisfactory results. GS can achieve comparable performance with respect to the other two strategies as the length of n-grams increases and for low threshold values (e.g. 0.4).

The experiments highlighted that GS generates lower anomaly scores compared to the other strategies. This potentially leads GS to produce fewer false positives

with a greater number of false negatives. On the other hand, LS and 2LS impose a more restrictive constraint on the payload structure and are more effective in detecting attacks whose contents share a considerable number of n-grams with the legitimate traffic.

Moreover, in conditions where the traffic is heterogeneous it is needed to raise the anomaly score threshold to avoid large false positive rates. In these cases GS would produce the highest number of false negatives. In addition, GS is less resistant to blending because its detection rate degrades more steeply with the threshold.

As far as blending is concerned, the worst case happens when a legitimate sequence of known n-grams is put at a position which is consistent with the normal traffic profile. To enhance robustness to blending, we introduced a factor of randomness concerning the model building process. For each distinct traffic profile PCkAD generates a random offset which is used to determine the byte starting from which the relevant payload of a packet is examined. Then each packet payload is treated as a circular array.

4.8 Considerations on encrypted traffic

HTTP is the protocol that enables communications between a user's browser and a website. The communications are in 'plain text', therefore any malicious user that manages to break into an HTTP connection can read its content. This scenario is very dangerous in that sensitive data such as credit card details or social security number can be stolen. In order to overcome the privacy and security issues arisen from HTTP protocol, cross-industry started to promote the use of HTTPS [109], the secure version of HTTP. HTTPS makes use of the TLS or SSL protocol to encrypt the communication channel. Historically, HTTPS connections were primarily used for payment transactions on the World Wide Web, e-mail and for sensitive transactions in corporate information systems. In the late 2000s and early 2010s, HTTPS began to see widespread use for protecting page authenticity on all types of websites, securing accounts and keeping user communications, identity and web browsing private.

Over the last few years HTTPS adoption has grown substantially. According to [76] the majority of desktop browsing is done over HTTPS and the number of top websites featuring this protocol has doubled between early 2016 and early 2017. Nonetheless, as of February 2017 about half of top websites still served HTTP by default and there were a lot of servers that did not support HTTPS at all.

The benefits provided by the increasingly widespread use of encrypted traffic are counter balanced by serious disadvantages. Encryption creates a problem for security because tools such as intrusion detection systems, anti-malware and data-loss prevention (DLP) solutions can no longer inspect the traffic, due to its obfuscation. This is a hard blow for defenders against malware and network intrusions. Thanks to encryption, crackers do not have to obfuscate malicious payloads in order to avoid drawing unwanted attention. They can focus on exploiting the vulnerabilities of their target system and enjoy a built-in masking against security solutions.

The security tools that we mentioned are of utmost importance and they have to be employed and PCkAD falls into this category. Our technique inspects application level traffic, therefore it can not achieve its goal when data are encrypted. A possible solution to keep using PCkAD is to decrypt the traffic, a solution that has already been adopted by other tools. Similarly to the next-generation firewalls from Palo Alto [5], when it comes to process inbound traffic to an internal web server or device, PCkAD could decrypt the traffic by knowing the SSL server certificate and key of

the target device. These data has to be provided by the device administrator. The decryption would not have any negative consequences on the network traffic, the only notable difference would be that PCkAD would be able to inspect packet data and detect malicious content over secure channels.

4.9 Summary

In this chapter, we described a novel anomaly-based intrusion detection technique, called PCkAD which achieved very good performances in detecting content-based attacks threatening application level protocols. PCkAD models legitimate traffic on the basis of the spatial distribution of the $n-$grams occurring in the relevant content of normal traffic. Separate models are built for different traffic lengths in order to better capture the relationship between size and content inherent in the network traffic. The technique achieved state of the art performances in real attack scenarios and is resistant to blending evasion techniques in that evading it is an intrinsically difficult problem.

Chapter 5

An approach to compress n-gram-based models for novelty detection

In the previous chapter we presented an intrusion detection technique designed to address a significant downside of n-gram-based approaches and to be resistant to blending attacks. In this chapter we further investigate the usage of the n-gram technique as a tool for developing a classification strategy, but with a different perspective. We recall that a n-gram g is a sequence of n symbols that belong to a given alphabet Σ, i.e. $g \in \Sigma_n$. n-grams are extracted from an input flow by resorting to a sliding window of length n. Excellent results have been achieved, even with 1-grams, but in general the best results are achieved with high order n-grams, i.e. n-grams with $n > 1$, most of the time even $n > 2$. Using such n-grams has the following implication: the greater n, the exponentially greater is the n-gram domain. For instance, in the context of the network intrusion detection, where typically byte values constitute the alphabet Σ, with just $n = 3$ the total number of n-grams that can be observed is over 16 million and with $n = 4$ there are over 4 billion. Depending on the approach that uses the n-gram technique, new problems may emerge, including: *i)* difficulties in storing all needed n-grams and *ii)* difficulties in building a classification model that generalises well the training data.

Another important thing to consider is that the greater the n-gram domain, the more likely is to observe similar n-gram distributions within the data. There might also be n-grams that carry very similar information, so that there is a certain degree of redundancy in a model. All these considerations lead to the following question: is it possible to renounce to a subset of the n-grams that are part of a classification model and get a new model with less n-grams but still with a satisfactory classification accuracy? This is the problem that we intend to explore in this chapter.

We propose an approach to compress n-gram-based classification model, which takes advantage of the k-center problem to select a representative subset of all the n-grams of a given model. Our goal is to assess whether it is possible to reduce the size of such models and still be able to classify data in a satisfactory way. This work has been conducted in the context of intrusion detection, indeed it follows a work concerning the development of a n-gram-based intrusion detection technique. We focus on content-based attack classification, a class of attacks for which the n-grams have proven to be very effective (see Section 4.3 for more details). We conducted a number of experiments on a real dataset and the preliminary results show that even if we retain only 5% of the model's n-grams we still get an $AUC > 95\%$.

The rest of the chapter is organized as follows. Section 5.1 provides an overview of the related work. Section 5.2 discusses the approach that we employ to build compressed models. Section 5.3 explains how we perform the classification phase.

Section 5.4 describes the experiments conducted to evaluate the compressed models built by means of our approach. Finally, Section 5.5 summarises the work.

5.1 Related work

The problem of compressing n-gram-based models has been addressed in many works, especially in the research field of speech recognition. In this context, a n-gram symbol corresponds to a word, so a n-gram represents a word sequence of length n. Several techniques have been proposed in the past, which can be categorised in *pruning*, *quantisation* and *world-class modelling* techniques. The former category is about training a model on all available training data and afterwards removing n-grams by pruning the model. Examples of strategies include removal of n-grams with low frequency counts [116] and entropy-based pruning [186]. The second category of techniques (e.g.[206] and [209]) apply a quantitation process in order to reduce the size of a n-gram-based model. Such process consists of mapping a variable with continuous range of values onto a discrete one, whose values belongs to a set of quantitation levels. The approach exploits the fact that many words sequences may have the same frequency and that the number of unique probabilities is considerably smaller than the total number of the n-gram probabilities in the model. World-class modelling techniques [42] [127] [95] seek to discover relationships between words and ultimately reduce the size of the vocabulary of the language model.

In the context of the intrusion detection, in [120] the authors propose a framework for network traffic classification and visualisation. They process web server textual log files to model HTTP queries with n-grams. Afterwards they derive a feature matrix from the n-gram frequency counts and reduce the number of dimensions by employing first PCA then Diffusion Map. At last, they use clustering to detect intrusions and visualise data. The latter work indirectly reduces the number of n-grams, by performing dimensionality reduction on the numerical representation of the HTTP queries, derived from the n-gram analysis. In contrast, our solution not only takes into account the frequency of the n-grams, but also the n-grams themselves to determine the fraction to use. In [130] the n-grams are used to detect malicious executables. The authors propose to combine the n-gram technique with a number of data mining and machine learning approaches, such as Bayes, decision trees and SVMs. Instead of using all the n-grams learned from the training data, the authors select only the most relevant fraction of the n-grams. To find the fraction, the authors analyse how the performances of the approaches vary with respect to the number of selected n-grams, the size of n-grams and the type of n-gram symbol (e.g. one byte, two bytes, etc). While the authors of [130] resort to an empirical approach to determine the fraction of n-grams, we propose a systematic approach that is supported by an extensive experimental campaign which allows us to understand how to tune the resulting classifier to get the best results.

5.2 Problem statement

Let T be a training set containing network packets. Our goal is to build a classification model M based on n-grams, such that it only contains a small fraction of all the n-grams observed in the training set.

As proposed with PCkAD, we partition the training set based on four criteria, in order to identify different types of legitimate traffic profiles. Such partition allows us to better capture the relationship between size and content inherent in the network

traffic and build more accurate models. The criteria used are the application protocol, e.g. `HTTP`, the direction (incoming or outgoing) and the specific application-level payload type. See Section 4.4.1 for an in-depth explanation.

In the light of the above discussion, we build a classification model for each identified network traffic profile. However, for the sake of explanation we will refer to only a single model M and the network traffic profile it describes, denoted as T_P. In order to build M we devised a process that consists of the following steps:

- training;
- compression;
- post training.

These steps are described in the following subsections.

5.2.1 Training phase

The objective of this phase is to build a model that describes the training packets of the traffic profile T_P, based on all the n-grams observed. We refer to this model as the *original model* M_o.

We learn the n-grams from the relevant payload of the network packets as PCkAD does. The relevant payload is defined as the portion of a packet that is actually exploited by an attacker to deliver malicious code (see Section 4.4.1 for additional details).

Let Ω_P be the set of packets occurring in T_P. Given a packet $p \in \Omega_P$, let p_r be its relevant payload and Σ_{p_r} be the set of n-grams occurring in p_r.

For each n-gram $g \in \Sigma_{p_r}$, we compute the relevant frequency f_g which is defined as

$$f_g = \frac{O(g, p_r)}{l_P} \tag{5.1}$$

where $O(g, p_r)$ is a function that returns the number of occurrences of g in a relevant payload p_r and $l_P = \max_i(|p_{r,i}|)$, with $p_{r,i} \in p_i$ and $|p_{r,i}|$ the length of $p_{r,i}$.

M_o is then a vector of length N where each dimension corresponds to a different n-gram and the $i{-}th$ element is the relative frequency f_i of the $i{-}th$ n-gram observed in T_P.

5.2.2 Compression phase

Data: M_o, k
Result: M_c
1 determine the set H of the most frequent n-grams in M_o;
2 randomly pick $z \in H$ and set $T = z$;
3 **while** $|T| < k$ **do**
4 compute $C = \{g \in M_o | g = argmax_{s \in M_o} \min_{t \in T} \rho(s,t)\}$;
5 pick $z \in C$ such that $z = argmax_{s \in C} f(s)$;
6 $T = T \cup \{z\}$;
7 **end**

Algorithm 1: The compression phase.

In this phase we take the output of the previous phase, i.e. M_o, and we build a new model M_c, called *the compressed model*. M_c contains only a fraction of the n-grams in M_o, which are chosen so that they can properly represent all the original

n-grams. Indeed, even though we are discarding n-grams, we want to limit the amount of information that is lost so that we can achieve satisfying classification accuracy.

In order to accomplish the goal we employ a clustering algorithm. Indeed clustering is commonly used to approximate a large/infinite/continuous set of data by a finite set of representatives. A well-known example of such usage is represented by vector quantization in audio processing.

Specifically, we resorted to the k-centre problem, which is described in the following. Given a metric space (Φ, ρ), where Φ is the set of data and ρ is a distance function that satisfies the three properties of a metric, let $S \subset \Phi$ be a finite set of data and k an integer, the problem asks to find the smallest radius r such that S is contained within k balls of radius r. The k-centre problem is NP-hard, thus we cannot count on an efficient algorithm that can always return the right solution. We then used a well-known greedy algorithm called *Farthest-first traversal* [94] . The algorithm takes S and k as input and builds a solution T one point at a time. The algorithm starts by choosing any point from S, and then iteratively picks the point furthest from the ones chosen so far. Farthest-first traversal is quite efficient, in fact its computational complexity is $O(k|S|)$. Moreover, the algorithm always returns a solution that is close to the optimal one, when the latter cannot be reached (see [94] for additional details).

We performed clustering in the metric space of length-n strings, the n-grams, with the *Hamming distance*. In our context the set S is represented by the set of n-grams that belongs to M_o, while k is chosen according to the percentage of n-grams that we want to retain. For example, given a model containing 10,000 n-grams, we might decide to retain 5% of the n-grams, therefore $k = 500$.

At last, we slightly modified the algorithm so that the selection of a centre is also influenced by the relative frequency of the n-grams. As shown in algorithm 1, we randomly choose a n-gram from the set H of the most frequent n-grams as initial point (lines 1 an 2). During the iteration process, we first determine the set C of candidate centres (line 4), i.e. the points that maximises the string-based distance with respect to T. Subsequently, we pick the candidate which has the highest frequency among all the candidates, where $f(s)$ is a function that returns the frequency of a n-gram s (lines 5 an 6). The reason behind this choice is bind to how we perform the classification, therefore it will be explained later, in Section 5.3.

5.2.3 Post training phase

Data: M_o, M_c, Ω_P^r
Result: M
initialise Θ;
initialise M;
for ***each*** $g \in M_o$ **do**
  compute $C_g = \{c \in M_c | c = argmin_{s \in M_c} d(g, c)\}$;
  for ***each*** $c \in C_g$ **do**
    pick $\theta_c \in \Theta$;
    $\theta_c = \theta_c \cup \{g\}$;
  end
end
for ***each*** $\theta \in \Theta$ **do**
  for ***each*** $p_r \in \Omega_P^r$ **do**
    compute $D_{p_r}^{\theta}$;
  end
  compute $D_{\Omega_P^r}^{\theta}$;
  add $< centre(\theta), D_{\Omega_P^r}^{\theta} >$ to M;
end

Algorithm 2: The post training phase.

At the end of the previous phase, we get a model M_c whose n-grams should represent all the n-grams in M_o. We believe that in order to perform a reliable classification, each centre c should tell us how the n-grams that it represents are distributed in the legitimate relevant payloads. Therefore, M_c cannot be used for the classification phase yet.

This phase takes as input both M_c and M_o and produces a new model M in output, which contains the same n-grams of M_c, each of which describes the distribution of the n-grams it represents.

Algorithm 2 shows the steps that are performed during the post training phase. We first need to perform a new clustering phase, this time with the goal of assigning each n-gram in M_o to its closest centre. The clustering is performed by combining two types of distances, i.e. the Hamming and a frequency-based distances. The resulting distance $d(g, c)$ between a n-gram g and a centre c is defined as:

$$d(g,c) = \begin{cases} d_S(g,c), & if \frac{|f_c - f_g|}{f_g} < t_f \\ +\infty, & otherwise \end{cases} \tag{5.2}$$

where $d_S(g, c)$ indicates the string-based distance and t_f is a frequency threshold.

Because we use a distance that incorporates a string similarity notion, during the search process we might end up finding multiple centres that are evenly close to a given n-gram g. We then initially evaluated two strategies to build the desired clusters, i.e. *i)* each n-gram $g \in M_o$ is assigned to one centre exclusively, based on some criteria; *ii)* each n-gram $g \in M_o$ is assigned to the closest centres. After preliminary investigations, we decided to discard the first option, because we could not find any suitable criteria to choose the right centre. We realised that by choosing a centre instead of another would lead to completely different results in some cases, which does not make sense to us. Regarding the second option, the aforementioned problem does not exist because we assign g to multiple clusters. Lines 4 to 8 show the clustering process, where the set C_g contains the closest centres to g and Θ is the set of clusters that we want to build.

Once Θ has been built, we proceed by computing the distribution of their n-grams within the legitimate relevant payloads of the network traffic profile (lines 12 to 14). First, we determined the set Ω_P for the traffic profile T_P described by M_o and derive the set of relevant payloads , which is defined as $\Omega_P^r = (p_r | p_r = relevant(p), with p \in \Omega_P)$, where *relevant (p)* is a function that returns the relevant content of a packet p. For each $\theta \in \Theta$ and for each $p_r \in \Omega_P^r$ we compute the following score:

$$D_{p_r}^{\theta} = \sum_{g \in \Sigma_{p_r}} O(g, p_r) \cdot \frac{|\Sigma_{p_r}^{\theta}|}{|\Sigma_{p_r}|} \tag{5.3}$$

The score is computed by summing the number of occurrences in p_r of every n-gram $g \in \Sigma_{p_r}$, such that $g \in \theta$, and multiplying it by the ratio of the distinct number of assigned n-grams, denoted as $|\Sigma_{p_r}^{\theta}|$, to the total number of distinct n-grams within the payload $|\Sigma_{p_r}|$.

After every $p_r \in \Omega_P^r$ has been inspected we compute the final score (line 14) which represents how the n-grams are distributed in Ω_P^r with respect to the centre of θ:

$$D_{\Omega_P^r}^{\theta} = \frac{\sum_{p_r \in \Omega_P^r} D_{p_r}^{\theta}}{\sum_{p_r \in \Omega_P^r} \frac{|\Sigma_{p_r}^{\theta}|}{|\Sigma_{p_r}|}} \tag{5.4}$$

The final step of this phase is to built the vector representing M where the i_{th} dimension is associated to a centre c_θ whose value is the pair $< f_i, D_{\Omega_P^r}^{\theta} >$ (line 15, where $centre(\theta)$ is a function that the returns the centre of θ).

5.3 Classification

The objective of this phase is to use a classification model M to compute the anomaly score of a relevant payload p_r. We want the anomaly score to reflect the similarity between the n-grams distribution within p_r and the distribution encoded in the centres that belongs to M.

The computation consists of two phases: *sp1)* n-grams distribution computation; *sp2)* anomaly score computation.

At the beginning of phase *sp1*, for each centre $\theta \in M$ we compute the score $D_{p_r}^{\theta}$. The result of this operation is a vector V containing the score of the centres that represent at least one n-gram $g \in \Sigma_{p_r}$. To conclude this phase we assign to each n-gram g an anomaly index $as_V(g)$ which is determined based on two different strategies, called *Brave* and *Cautious*, and computed as:

$$as_V(g) = op_{\theta \in \Theta_g} V(\theta) \tag{5.5}$$

where Θ_g is the set of the closest centres to g and op is an operation that is determined by the chosen strategy, i.e. *min* or *max* when the *Brave* or *Cautious* strategy is chosen, respectively. We expect to observed a difference between the distribution of n-grams in malicious payloads and the distribution represented by the model that is significantly higher than the difference between n-grams in legitimate payloads and the model. However, there might exist a considerable number of n-grams in malicious payloads for which there exists a centre with a very similar distribution or a considerable number of n-grams in legitimate payloads such that there exists a centre with a very different distribution. Such n-grams may lead to the computation

of anomaly scores that poorly represent the true nature of a payload. In order to mitigate this negative influence, if the number of the first type of n-gram is significantly greater then we employ the *Cautious* approach, otherwise the *Brave* approach.

The anomaly score is then computed in the second phase *sp2*. To achieve this goal we introduce the concept of *byte coverage*. Given a relevant payload p_r, let B_{p_r} be its set of bytes. For each byte $b \in B_{p_r}$ we determine the set of n-grams that contain b, denoted as G_b. After that, we compute a score for b, here referred to as $as_{G_b}(b)$, as a function of the scores associated to the n-grams in G_b. We evaluated three criteria to compute $as_{G_b}(b)$, i.e. we select *i)* the minimum score in G_b, *ii)* the maximum score in G_b or *iii)* the average score in G_b. The main idea behind the first criterion is to trust the fact that anomalous payloads contain n-grams whose distribution is sufficiently different from that encoded in its closest centres, if any. Concerning the second criterion, we rely on the fact that normal payloads contain n-grams whose distribution is very similar to that encoded in its closest centres.

It is important to note that there might exist one or more n-grams $g \in \Sigma_{p_r}$ such that $\Theta = \emptyset$, so it is possible to observe at least one byte b such that $G_b = \emptyset$, i.e. there is no coverage. In this case, we set $as_{G_b}(b)$ to 1, the maximum score possible. This strategy allows us to properly weight the presence of n-grams that cannot be assigned to any centres. By picking the most frequent n-grams during the compression phase, we might be able to get a better coverage, especially for legitimate payloads. This strategy might be beneficial to the performances of our approach, in that the anomaly score assigned to a legitimate payload should be lower than that assigned to an anomalous payload. To conclude, the anomaly score $as(p_r)$ of p_r is computed as:

$$as(p_r) = \frac{\sum_{b \in B_{p_r}} as_{G_b}(b)}{|B_{p_r}|} \tag{5.6}$$

where $|B_{p_r}|$ is the total number of bytes.

5.4 Experimental validation

In this section we assess the performances of our approach. We build a variety of compressed models and for each of them we compute the *Area Under the Curve* or *AUC*, which is a classification measure that reflects the probability that the classifier will compute a higher score for a randomly chosen positive example than for a randomly chosen negative example. Such models are built based on different parameter configurations, which are discussed in the next section. Experiments were conducted on the *UW* dataset, previously introduced in Chapter 4. However, this time we used a subset of the dataset, that was obtained by selecting only the traffic profiles for which there are both normal and malicious packets. We chose to do so because we are mainly interested in knowing how effective the compressed models are to distinguish between normal and malicious packets. In the case of PCkAD, the profiles for which there are no malicious packets are still meaningful because they could be used whenever a test sample does not match any existing profile. So in this case we resort to the concept of model similarity to find a similar traffic profile to compute the anomaly score (see Section 4.5 for additional details).

5.4.1 Parameter setting

In order to generate a compressed model M there are a few parameters that need to be set, i.e.: cl, crt_n, crt_b, th_s, th_f and n. The parameters represent aspects that were discussed in the previous sections. cl denotes the compression level that we desire for M, specifically if $cl = 5\%$ it means that M contains just 5% of the number of n-grams that belong to its corresponding original model M_o. This parameter is used to set the parameter k that is used to solve the k-centre problem in the compression phase. k is computed as $k = |M_o| \cdot \frac{cl}{100}$, where $|M_o|$ is the number of n-grams contained in M_o. For the experiments we chose values that range from 5% to 30% with a step of 5%.The parameter crt_b represents the criterion that is used to assign an anomaly index to each payload byte, during the classification phase. It can take only three values, i.e. min, max and avg. The parameter crt_n is used to choose between the *Brave* and *Cautious* strategies. Both th_s and th_f are thresholds, where the former is used for string-based distances and the latter for frequency-based distances. The parameters are used in both the post training and classification phase. th_s determines the similarity between n-grams and we chose the range of values from 1 to $n-1$, to evaluate the impact of this parameter on the performances of M. Regarding th_f, preliminary results showed that it has a little impact on the performances. This is due to the fact that the majority of high order n-grams share the same distribution in the dataset we used, therefore only a small portion of them is affected by th_f. Given the aforementioned results, we decided to not consider th_f in our analysis. The last parameter mentioned is n, i.e. the n-gram length, for which we explored the values $(3, 4, 6, 10)$, so that we could observe how the performances of M change with respect to small and large high order n-grams. Before presenting the results, we split the traffic in profiles, similarly to PCkAD. The traffic consisted of incoming HTTP packets and was split based on the type of packets, i.e. GET and POST, and the range of length. With regard to the latter criterion, a subset of packets is formed when the difference between the relevant payload length of any pair of packets is less or equal to 20 bytes. The value was chosen based on the results presented in the previous chapter.

5.4.2 Results

Figures 5.1 and 5.2 report the experimental results for $crt_n = Brave$, while 5.3 and 5.4 for $crt_n = Cautious$. The Y axis represents the AUC and the X axis the compression level cl. We generate a distinct AUC curve for each value of th_s and a dedicated graph for each value of crt_b. We note that with $n = 6, 10$ the number of curves gets too large, so for the sake of visualisation we decided to show only the best four curves per graph. For each combination of crt_n and crt_b, in Table 5.1 we present the average AUC of the best curve, with the purpose of highlighting the stability of the different configurations. Moreover, Tables 5.2 and 5.3 report the highest AUC values observed in each graph and for each AUC the compression level is specified between parenthesis.

In the following we discuss the results. For some configurations we achieve an AUC greater than 95% (fig. 5.3 (b, d, f), 5.1 (b, d, f) and 5.2 (f)) and there are a few other configurations that allow us to achieve an AUC between 90% and 95% (fig. 5.1 (a, c, e), 5.2 (a, b, d, e, f), 5.3 (a, c, e) and 5.4 (a, b, c, d, e, f)) . Even more surprisingly, the best results in some cases are provided by the configurations with the lowest compression level, i.e. $cl = 5\%$. With $n = 3, 4$ the best performances are achieved

TABLE 5.1: Average AUC for each combination of crt_n and crt_b.

crt_n	Brave			Cautious		
crt_b	Min	Max	Avg	Min	Max	Avg
n = 3	0.8840	0.8488	0.8702	0.7433	0.7283	0.7316
n = 4	0.9469	0.9467	0.9481	0.9484	0.9470	0.9515
n = 6	0.8237	0.8640	0.8368	0.8079	0.8911	0.8919
n = 10	0.7899	0.8872	0.8447	0.8272	0.9098	0.9005

when we select the lowest th_s possible, in other words we demand that the difference between a centre and a payload n-gram lies in just one byte. The result holds for both the *Brave* and *Cautious* strategies, even though there is a difference between them. It seems that the former guarantees more stable performances than the latter, as can be seen by comparing figures 5.1 and 5.3. When $crt_n = Cautious$, the curve that achieves the best AUC value is characterised by values that drastically decrease by varying the cl value, especially for $n = 3$. Indeed, the average AUC values in Table 5.1 are greater for the *Brave* approach, even though the maximum AUC values observed are comparable. On the other hand, with $n = 6, 10$ the situation is the opposite, in that now we achieve the best performances when setting th_s to high values. This can be explained by recalling that the greater n the exponentially greater is the n-gram domain, therefore with $n = 6, 10$ there is such a huge variety of n-grams that it gets more and more difficult to assign many payload n-grams to the centres. As a consequence, the byte coverage of legitimate payloads should get worse and lead to higher anomaly scores. In this case, there is not a significant difference between the *Brave* and *Cautious* strategies, in terms of stability.

According to the results, when $n = 3, 4$ the best values to choose for crt_b are *min* and *avg*, while for the $n = 6, 10$ the option *max* is slightly better. As can be seen in Tables 5.2 and 5.3, the maximum AUC values are achieved with small compression levels, generally between 5% to 20%. However, note that for some configurations (e.g. $crt_n = Cautious$, $crt_b = max$ and $n = 4$) the difference between the smallest compression level and the one reported in the table is negligible.

5.4.3 Discussion

The experimental results previously discussed are promising. The approach is able to distinguish between normal and malicious packet payloads in a satisfactory way, with only 5% of the total n-grams that characterise the network traffic. In other words, the space required by the approach is very small and this has interesting implications in that it could naturally find its applicability in an IoT environment, where it could be deployed in resource-constrained devices [215]. We also believe that having smaller models to work with may lead to improving the processing speed of the approach. We would like to explore this aspect in the future.

Even though the dataset that we used for these experiments is a subset of the *UW* dataset used in Chapter 4, we believe it is worth to mention that there is a significant gap between the performances of the compressed models and those achieved by PAYL and Spectrogram, two IDS techniques that use all the learnt n-grams. The compressed models obtained an AUC which is greater than 95% while PAYL and Spectrogram about 83% and 86%, respectively.

TABLE 5.2: Maximum AUC for $crt_n = Brave$ by varying crt_b. For each AUC the compression level is specified between parenthesis.

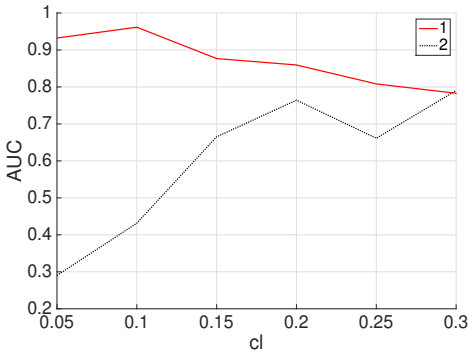
crt_b	Min	Max	Avg
n = 3	0.9609 (10%)	0.9482 (15%)	0.9615 (10%)
n = 4	0.9726 (30%)	0.9692 (20%)	0.9697 (20%)
n = 6	0.8654 (15%)	0.8960 (20%)	0.9009 (10%)
n = 10	0.9266 (15%)	0.9433 (5%)	0.9397 (5%)

TABLE 5.3: Maximum AUC for $crt_n = Cautious$ by varying crt_b. For each AUC the compression level is specified between parenthesis.

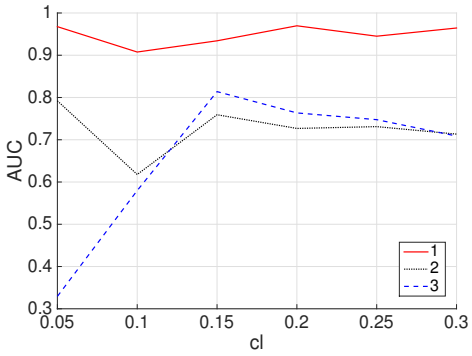
crt_b	Min	Max	Avg
n = 3	0.9395 (5%)	0.9312 (5%)	0.9351 (5%)
n = 4	0.9732 (30%)	0.9695 (20%)	0.9757 (20%)
n = 6	0.9197 (10%)	0.8959 (15%)	0.9278 (10%)
n = 10	0.8776 (5%)	0.9269 (5%)	0.9396 (5%)

5.5 Summary

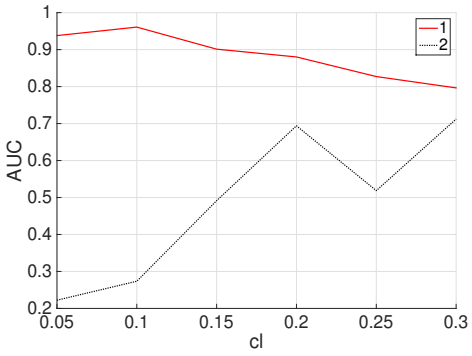
In this chapter we proposed an approach for compressing n-gram-based classification models, in the context of intrusion detection. Specifically, our goal was to build compressed models that can classify content-based attacks in a satisfactory way. We first build a model containing all the n-grams learned from the training data and then we compress it by resorting to the k-centre problem. At last, we perform an additional post training step to compute the n-gram distribution within the legitimate relevant payloads, for each centre. We introduced the concept of byte coverage to compute the anomaly score of a packet. Experiments were conducted on a real word dataset to assess the performances of compressed models. The experimental results show promising results, we achieved an AUC greater than 95% for significant parameter configurations.



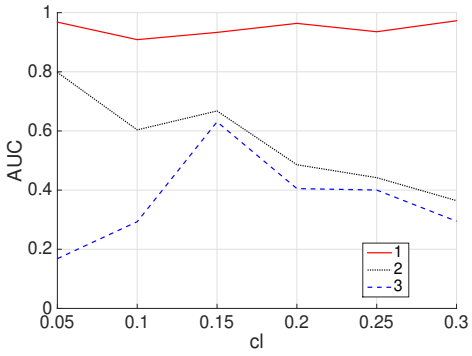
(A) $crt = avg$, $n = 3$.



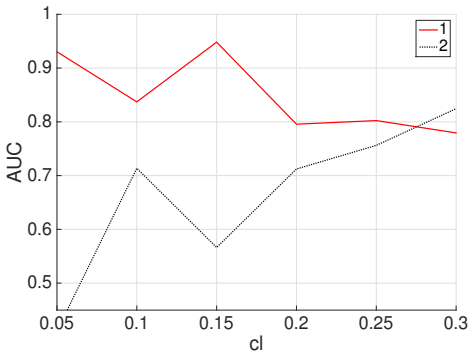
(B) $crt = avg$, $n = 4$.



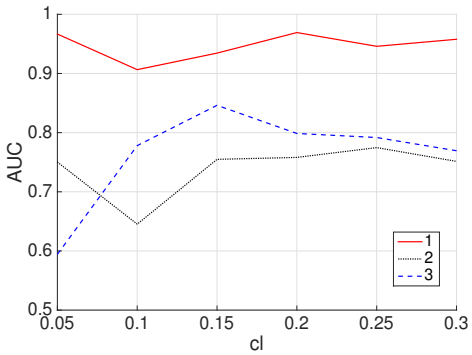
(C) $crt = min$, $n = 3$.



(D) $crt = min$, $n = 4$.

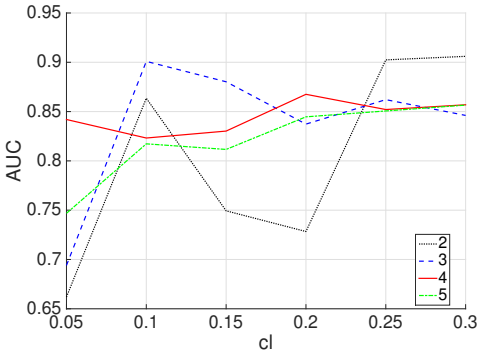


(E) $crt = max$, $n = 3$.

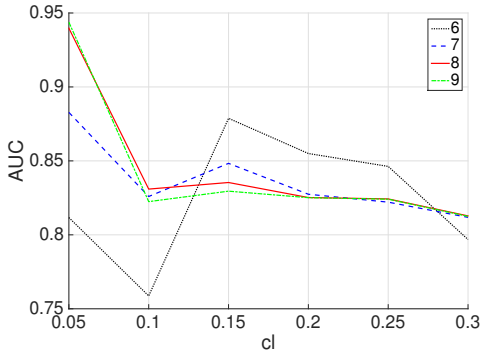


(F) $crt = max$, $n = 4$.

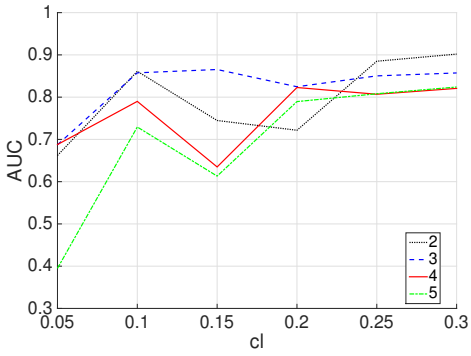
FIGURE 5.1: Results for $n = 3$ and $n = 4$ with the *brave* approach.



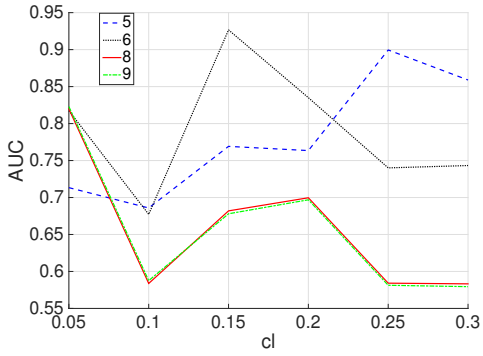
(A) $crt = avg$, $n = 6$.



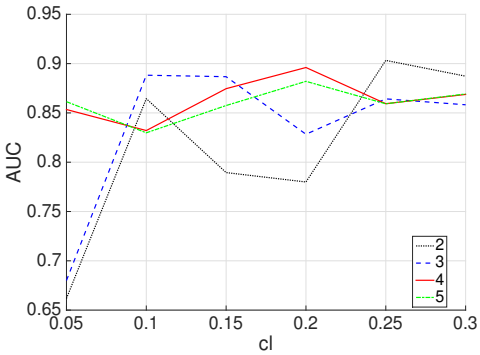
(B) $crt = avg$, $n = 10$.



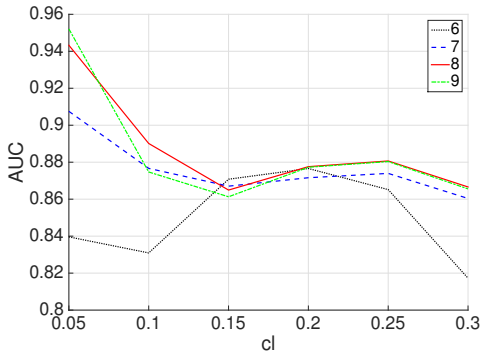
(C) $crt = min$, $n = 6$.



(D) $crt = min$, $n = 10$.

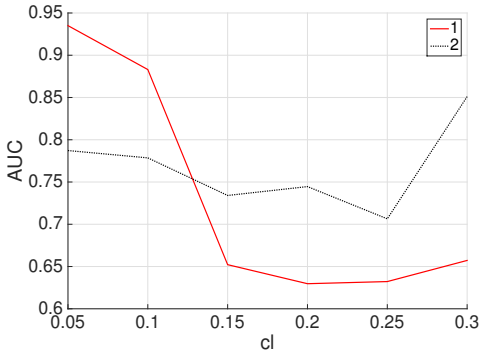


(E) $crt = max$, $n = 6$.

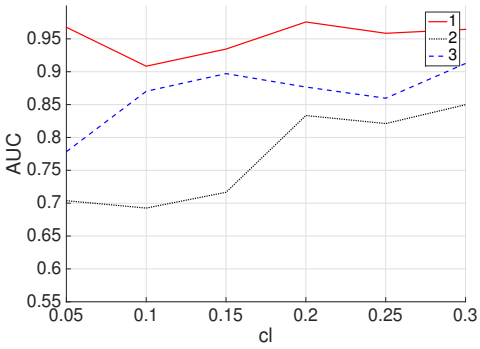


(F) $crt = max$, $n = 10$.

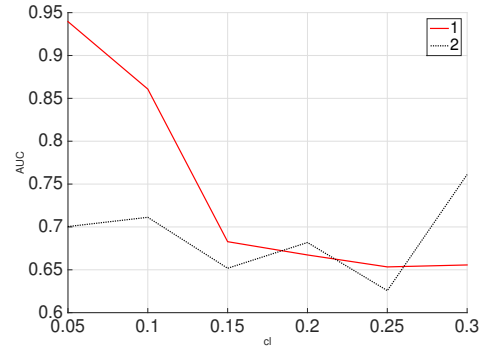
FIGURE 5.2: Results for $n = 6$ and $n = 10$ with the *brave* approach.



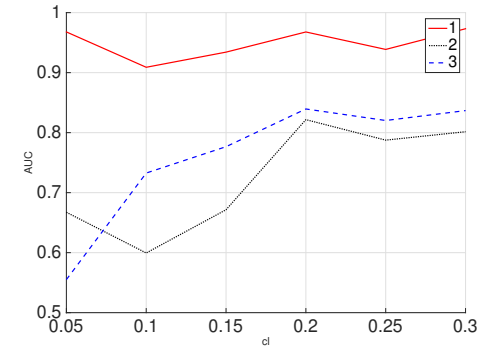
(A) $crt = avg$, $n = 3$.



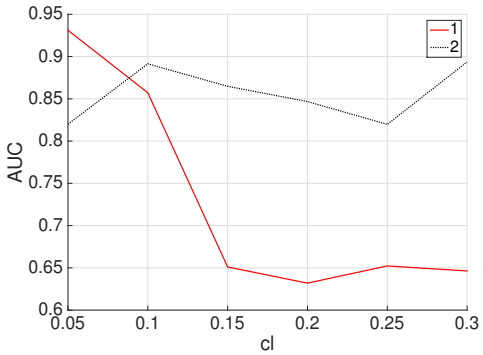
(B) $crt = avg$, $n = 4$.



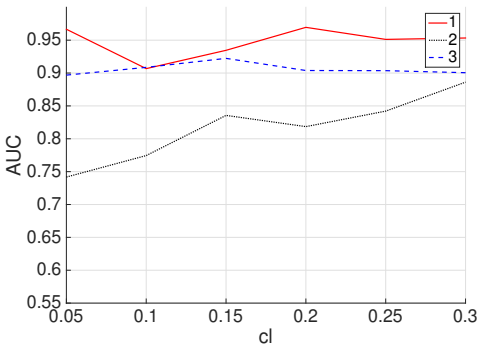
(C) $crt = min$, $n = 3$.



(D) $crt = min$, $n = 4$.

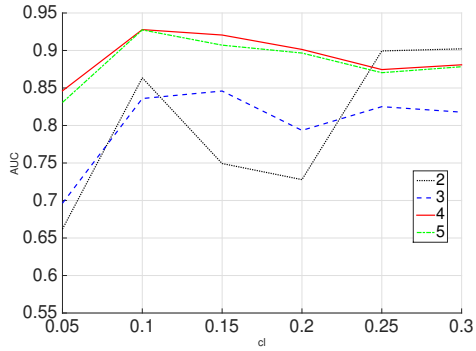


(E) $crt = max$, $n = 3$.

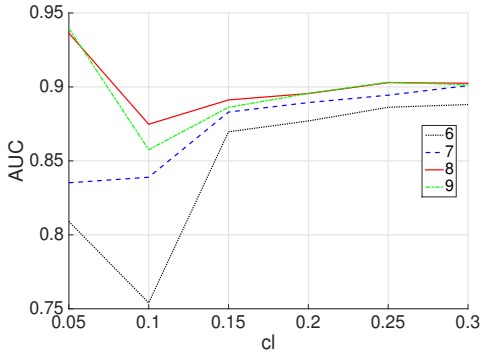


(F) $crt = max$, $n = 4$.

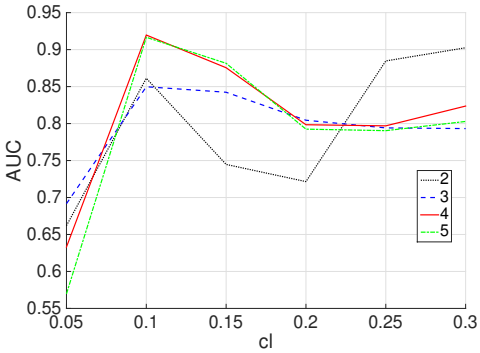
FIGURE 5.3: Results for $n = 3$ and $n = 4$ with the *cautious* approach.



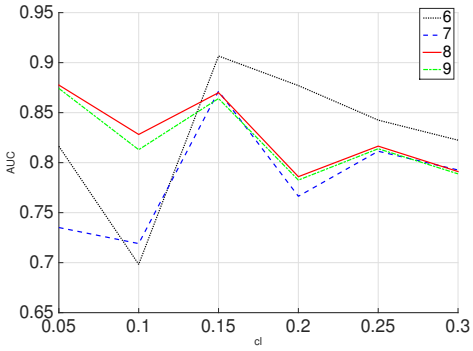
(A) $crt = avg$, $n = 6$.



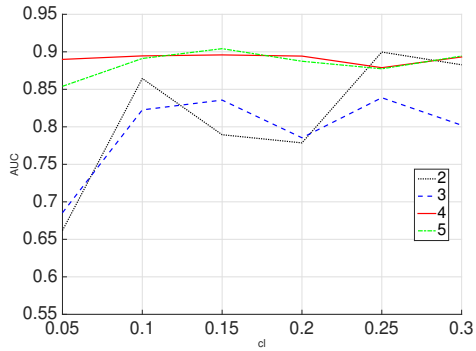
(B) $crt = avg$, $n = 10$.



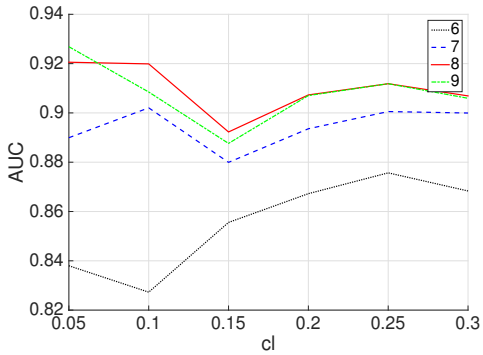
(C) $crt = min$, $n = 6$.



(D) $crt = min$, $n = 10$.



(E) $crt = max$, $n = 6$.



(F) $crt = max$, $n = 10$.

FIGURE 5.4: Results for $n = 6$ and $n = 10$ with the *cautious* approach.

Chapter 6

Adaptive Access Control with Machine Learning

In the previous chapters we discussed about anomalies that are interesting to detect because they represent intrusions in a network or system. In this chapter we consider anomalies that are attempts of exceeding or misusing access controls to negatively affect the confidentiality, integrity or availability of a target information system.

Nowadays computing systems are at the basis of our daily activities, enabling always more and more services and management of critical applications. The key ingredient of all such modern computing systems is the data: controlling accesses to data is indeed of paramount importance. Access control systems are the first line of defence for data as they establish fine-grained conditions under which allowed users can act. Multiple approaches to access control have been proposed, each of which enjoys different properties. However, all of them are intrinsically static: access control is not conceived to adapt over time thus to evolve according to the behaviours occurred in the controlled system.

The recently proposed Attribute-Based Access Control (ABAC) [103] is a flexible and expressive approach which bases its access decisions on so-called *attributes*, any information available from the evaluation context such as subject's identifier and role, resource's type, current time and location data. State-of-the-art ABAC infrastructure, such as the XACML standard [163], divides calculation and enforcement of access decisions enabling support for the introduction of new attributes at runtime. However, maintenance and administration of access control policies are still carried out in a manual fashion. The policies in force at a given instance are the result of the information available at design phase, they can not address access needs emerging at runtime [107]. This lack of adaptation makes access control systems weaker and weaker over time: authorised users may figure out allowed access patterns to perform insider threat such as disclosure or stealing of sensitive data. Therefore, ABAC systems lack of timely adaptation means to dynamically changing policies according to occurred behaviours.

As a matter of fact, weaknesses of access control systems not spotted in time are always causing huge economic losses to companies or international embarrassment to public bodies. Just to mention a few, an US soldier took advantage of granted access rights to dump a massively amount of classified data then breached on WikiLeaks[1]. Similarly, a major Intel microprocessor project was disclosed to a competitor by a resigned employee whose managed to maintain access to classified projects

[1] https://www.csmonitor.com/USA/2011/1222/Bradley-Manning-case-signals-US-vulnerability-to-insider-cyberattack

while he was working for the competitor [2]. More and more of such cases happen daily, but adequate countermeasures can be put in place to monitor and timely spot malicious behaviours. Indeed, both cases would have been avoided by a runtime refinement mechanism for access control policies: behavioural features like writing/reading patterns, timing and location of access represent a normal user conduct that if properly encoded within policies would avoid by-design such insider threat attacks.

Detecting and preventing insider threats by monitoring and evaluation access control policies has been an active aerea of research. Common approaches rest on rule mining techniques to discover harmful exploitable policy faults [108, 25]. Alternatively, behavioural models have been proposed [166, 147, 138], both online and offline solutions, to detect based on monitoring system insider threats. However, both solutions are just detecting threats or faults and do not propose any knowledge-informed adaptation to the access control policies so to timely avoid such malicious behaviour. An attempt to adapt access control policies is described in [102]. However the approach ignores valuable contextual knowledge and do not build behavioural models.

The highly dynamic environment where access control systems work can also cause additional issues: access policies become out-of-data soon and would require continuative maintenance to preserve adequate access rights to new controlled resources and for changing users' patterns. Indeed, unforeseen needed access rights could emerge over time from the monitored patterns, for instance users working in commercial Chinese Wall environments whose access rights are not properly set to carry out required business tasks. Generally speaking, this would amount to being able of detecting inconsistencies of (un)granted access so to automatically modify and adapt the access control policies by themselves.

In this chapter, we propose an approach based on machine learning to dynamically refine and update policies, respectively, to prevent insider threats and to automate policy administration. The proposed learning approach permits refinement and generation of new access control rules according to behavioural features monitored at runtime. The designed system, named ML-AC, exploits a white-box decision learning approach whose aim is to learn behavioural profiles of users accessing resources so to accurately refine policies and detect anomalies or inconsistencies for classes of users. Such behavioural profiles are here called *classes of interaction*.

Practically, ML-AC uses knowledge on access patterns learned at runtime to introduce behavioural rules into the policy both to avoid abuse of granted rights and fix inconsistencies. Behavioural rules are used to refine access policies with controls on features like frequency of access, amount of data, location, etc. Based on such behavioural profile, ML-AC will also introduce amendments to policies to harmonise access patterns.

Contributions. The main contributions of this work are:

- to introduce the system architecture of ML-AC which integrates ABAC with machine learning to autonomously refine and update access control policies;
- to tailor well-known machine learning techniques such as *decision tree* to automate generation and update of behavioural rules;
- to discuss a case study to show the capabilities of ML-AC and to conduct an experiment to assess the impact of the classes of interaction.

[2] http://www.insiderthreatdefense.com/pdfs/Insider%20Threats%20Incidents-Could%20They%20Happen%20To%20Your%20Organization.pdf

The rest of the chapter is organised as follows. Section 6.1 discusses the related work. Section 6.2 provides a brief description of how machine learning is used in this work. The architecture of ML-AC and its components are described in Section 6.3, while Section 6.4 details how machine learning is used to achieve our goals. Section 6.5 describes a case study that is used to show how the system works. Section 6.6 presents an experiment conducted for assessing the impact of the classes of interaction. Finally, Section 6.7 concludes the chapter.

6.1 Related work

This work is closely related to three main areas of research: insider threats detection, policy misconfiguration detection and adaptive access control.

Few works have addressed the problem of dynamically update access control policies based on the user exhibited behaviour. Hummer et al. [107] propose a dynamic policy management process (DPMP) for access and identity management. DPMP aims to discover new and potentially relevant policies as well as outdated policies that reflect the current system state. Such policies describe new relationships between users and resources that might be integrated in the current policies. In contrast, ML-AC aims to automatically update the employed policies by adding new controls that capture run-time aspects concerning access control. Similarly to ML-AC, Costante et al. [54] use an anomaly-based engine to automatically learn a model of normal user behavior, making it possible to flag anomalous transactions, and exploit an operator's feedback on alerts to automatically build and update signatures of attacks that are used to timely block suspicious transactions.

The closest work to ours is BBNAC [79]. BBNAC is a behaviour-based network access control system, where behaviour-based access control policies are employed instead of rule-based policies. The mechanism employs a clustering-based algorithm to perform access control. In its first incarnation, BBNAC required human intervention in order to perform its tasks. It was later enhanced in [80] to work in an automatic fashion. However, BBNAC suffers from the lack of static knowledge on users, operations and resources, only relying on run-time aspects. Therefore, differently from ML-AC, it can build models that do not properly distinguish between behaviours generated by different types of user, as we show in the experiments discussed in Section 6.6.

A number of approaches [108, 25] for detecting faults within a policy use association rule mining. Hwang et al. [108] mines likely properties (patterns of interest) from a policy under verification via association rule mining to learn relationships between subject, action, and subject-action attributes. The properties are verified by producing counterexamples that help policy makers to detect faults. The approach, however, does not take into account run-time user behaviours, thus limiting its detection capabilities to request-decision pairs encoded in the policy. Bauer et al. [25] instead aims to predict changes in access control policies that are likely to reflect users' intentions. These changes can represent potential misconfigurations that could interfere with legitimate accesses. However, as pointed out in Section 6.4.3, this approach can produce a significant number of wrong predictions due to the lack of contextual knowledge.

Following the realization of the risks posed by insider threats, several approaches for insider threat detection have been proposed in the last years. Maloof et al. [147] propose a system called *ELICIT* to detect insiders that violate need-to-know. The

system focuses on a variety of user activities, including searching, browsing, downloading, etc. It builds behavioural models from contextual information about past activities, user identities and social networks. A Bayesian network is then employed to rank insider threats. A more recent work [138] presents a system designed to analyze data repositories and activity logs to characterize recent activities performed by users within an organization. The system extracts a rich set of features from the observed user profiles and employs several anomaly metrics to score user behaviours. Alizadeh et al. [4] propose an approach for behavioral analysis to detect anomalous behaviors in the use of break-the-glass within a Dutch hospital. The underlying idea is to build histogram-based profiles representing user and group behaviors and compare those profiles to measure to what extent a user behaves differently from users having the same job functions. Those works, however, only detect anomalous behaviors based on contextual knowledge.

Other works have also exploited knowledge access control policies to detect insider threats. Hu et al. [102] propose an approach that combines *Role-Based Access Control* (*RBAC*) and genetic algorithms to generate role-action mapping rules for insider threat detection. The main assumption at the basis of this work is that users with similar job functions tend to use resources in a similar fashion. The authors focus on users, roles and permitted/expected processes to build descriptive models. Another work that exploits the knowledge of access control for insider threat detection is [166]. The authors propose a monitoring mechanism that analyses both role and individual behavioural profiles to detect coarse and finer-grained anomalies, respectively. The behavioural models are quite simple; they only describe the operations performed by the insiders (e.g., search, send, copy) and their frequency patterns. Because of such simplicity, the applicability of the extracted models might be limited. In contrast, ML-AC provides a system that neither relies only on the concept of role nor uses a fixed set of features, so it can be adapted to any application domain.

6.2 A machine learning approach for access control

In this section we explain how we employ machine learning (ML) to achieve our objectives. We build two different types of ML models: a model that describes how the users utilise the resources and a model that describes the users' resource access patterns. The former type of models are built by means of the Random Forest algorithm and are used to generate dynamic policy (or behavioural) rules. These rules are then integrated in the original security policy, so that they can be used for the detection of legitimate users with malicious intent. On the other hand, the latter type of models are built by resorting to association rule mining and are used to identify policy inconsistencies.

As anticipated in the introduction, the system models user behaviours based on the concept of class of interaction, where a class is denoted as C. The concept describes a set of user behaviours exhibited by one or more users that access to a specific set of resources by means of a selected set of operations. Their goal is to differentiate between different kinds of user behaviours to better detect legitimate users with malicious intent, therefore the behaviours contained in a class C_i are different from those contained in C_j. Such differentiation was previously proposed in [160], where the authors show that insider detection can be improved by distinguishing user behaviours related to different job roles. To achieve their objective they exploit the concept of role in a *Role-Based Access Control* system or *RBAC*. The

main assumption behind their work is that users with the same role generate similar behaviours. We do not limit the differentiation only to the concept of role, but we define the classes of interaction based on knowledge derived from a security policy and knowledge that concerns a specific application domain. The latter knowledge is exploited to define the behavioural features that characterise the user behaviours. In terms of machine learning, we achieve the differentiation by building a distinct behavioural model for each class of interaction.

In order to make it clear the benefits of the combination of machine learning with an access control system, in the following we discuss how their combination can be employed to detect anomalous behaviours.

A user behaviour is considered anomalous if it exhibits characteristics that are significantly different from those exhibited by the remaining population of user behaviours. However, there may be behaviours, generated by users that belong to a class of interaction C_i, that are *anomalous* with respect to those observed in C_i, but are *normal* with respect to another existing class C_j. This might be a problem in the context of insider threat detection, because a user might willingly act as another user to evade detection. However, it might also happen that a legitimate user that is performing a malicious action could unwillingly exhibit a behaviour that is similar to those of other classes of interaction. Those kind of behaviours might be hardly classified as anomalies, without making a proper distinction between the two sets of behaviours. If the knowledge available in the context of access control were exploited (e.g. types of users and resources) then the classifier would be able to recognise the type of user that is generating the anomalous behaviour and would use the correct knowledge in order to classify the behaviour. We refer to the anomalous behaviours that require the exploitation of contextual knowledge as *Sneaky Anomalies*, because it is particularly difficult to classify them otherwise. The other type of anomalies are referred to as *Outstanding Anomalies*. We use the notation b for a generic behaviour, b_i for a behaviour associated to C_i and b_i^j for a sneaky behaviour which is similar to behaviours from C_j but anomalous with respect to those from C_i.

Knowing the classes of interaction, the system can build a set of dynamic rules for each class in order to better represent their behaviours. Let C_1, ..., C_N be N classes of interaction, where each distinct C_i exhibits user behaviours that are different from those related to C_j, with $i \neq j$. ML-AC builds N different sets of behavioural rules, here referred to as B_1, ..., B_N. Suppose now that ML-AC has to classify an anomalous user behaviour b_i. Because ML-AC distinguishes among the different classes of interaction, it knows the origin of u_i and, as a consequence, that the best way to classify b_i is to use B_i. If the anomalous behaviour were sneaky, i.e. b_i^j, ML-AC would then use B_i to perform the classification, so that it is more likely to classify the behaviour correctly. On the contrary, if ML-AC modelled user behaviours related to different classes of interaction as if they were part of the same class, the anomaly might go undetected.

6.2.1 An example of security policy

```
Policy set policy1 {permit-overrides
      target: equal ("R 1", resource/type) policies:
Rule rule1(permit target:
            equal ("Read", action/action-ID)
            && equal ("Junior manager", subject/role)
            && equal ("Department 1", subject/department)
            && equal ("Project A", subject/project)
```

```
      )
Rule rule1(permit target:
            equal ("Write", action/action-ID)
            && equal ("Junior manager", subject/role)
            && equal ("Department 1", subject/department)
            && equal ("Project A", subject/project)
      )
}
```

LISTING 6.1: A portion of a ABAC policy π.

```
Policy set policy1 {permit-overrides
      target: equal ("R 1", resource/type) policies:
Rule rule1(permit target:
            equal ("Read", action/action-ID)
            && equal ("Junior manager", subject/role)
            && equal ("Department 1", subject/department)
            && equal ("Project A", subject/project)
            && less-than (feature/BytesRead, 345.6)
            && less-than (feature/NumberOfReads, 14)
            && greater-than (feature/BytesRead, 98.1)
            && greater-than (feature/NumberOfReads, 4)
      )
Rule rule1(permit target:
            equal ("Write", action/action-ID)
            && equal ("Junior manager", subject/role)
            && equal ("Department 1", subject/department)
            && equal ("Project A", subject/project)
            && less-than (feature/BytesWritten, 234.5)
            && less-than (feature/NumberOfWrites, 12)
      )
}
```

LISTING 6.2: A portion of a new policy (π_{ML}) containing behavioural rules.

```
Subject's "department"="Department 1"
Subject's "role"="Junior manager"
Subject's "project"="Project A"
Action="read"
Resource "type"="R 1"
```

LISTING 6.3: An example of user request.

```
Subject's "department"="Department 1"
Subject's "role"="Junior manager"
Subject's "project"="Project A"
Action="read"
Resource "type"="R 1"
Feature "BytesRead"="246.2"
Feature "NumberOfReads"="8"
```

LISTING 6.4: An example of request from a legitimate user that acts normally.

```
Subject's "department"="Department 1"
Subject's "role"="Junior manager"
Subject's "project"="Project A"
Action="read"
```

```
Resource "type"="R 1"
Feature "BytesRead"="591.6"
Feature "NumberOfReads"="22"
```

LISTING 6.5: An example of request from a legitimate user with malicious intentions.

To further highlight the role of machine learning in the context of access control, we now present an example of a portion of a ABAC policy π that is part of a case study that will be discussed in depth in Section 6.5. The portion reported in the listing 6.1 defines two rules that allow users that has the role of *junior manager* to access to resources of type R_1. Not all the junior managers have this privilege, only those that work on *Project A*, are from *Department 1* and perform read and write operations. The listing 6.3 describes a user request. As long as the attributes specified by the user satisfy the policy rules, the access control system will authorise the access, regardless of how the user behaves. If the user were performing an anomalous number of reads (e.g. to steal sensitive data in a short time window), the access control system would still authorise the access. We want the access control system to be able to recognise these scenarios and to accomplish this goal we exploit machine learning to model user behaviours. We aim at changing π by integrating dynamic rules that describe user behaviours and get a new policy π_{ML}, as shown in Listing 6.2. The attributes (*BytesRead*, *NumberOfReads*, *BytesWritten*, *NumberOfWrites*) represent behavioural features and are used to model user behaviours. After this integration, a user request looks like those reported in listings 6.4 and 6.5. The request contains both static and "behavioural" attributes. Note that only the former attributes are specified by the user, while the latter ones are provided by a monitoring system. The request 6.4 is generated by a user who is acting normally, in fact the values of the behavioural attributes satisfy the policy rules. On the other hand, the request 6.5 represents the attempt of a junior manager to steal sensitive data. In fact, both *BytesRead* and *NumberOfReads* have anomalous values. In this case, the access control system is able to recognise that a legitimate user is acting in a malicious way and, as a consequence, the access will be denied.

6.3 System model

The model of the proposed system is depicted in Figure 6.1, which shows the fundamental components of ML-AC and the most significant interaction among them. The three main components of ML-AC are *Access Control*, *Machine Learning* and

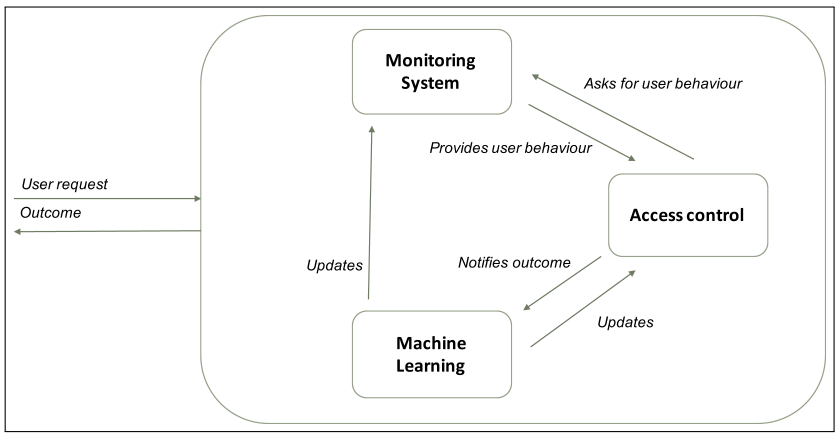


FIGURE 6.1: High-level architecture

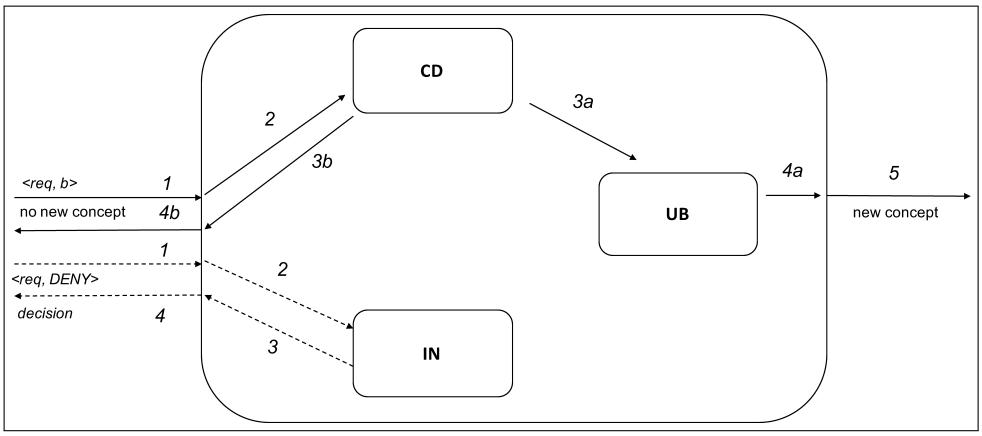


FIGURE 6.2: ML_c's subcomponents and interactions.

Monitoring System and are referred to as AC_c, ML_c and MS_c, respectively. ML-AC receives a user request req as input and produces a *deny* or *permit* as output. AC_c serves the purpose of evaluating the request, by resorting to both static and dynamic policy rules. Upon the reception of req, only static attributes are known (i.e. those specified by the user). In order to assess whether a legitimate user is behaving as expected, AC_c contacts MS_c, which is in charge of computing the user behaviour. MS_c returns the user behaviour to AC_c, which is now able to answer to the request. Note that MS_c should know the class of interaction of each behaviour so that it can properly monitor the users and keep their behaviours updated. After the evaluation of req, AC_c communicates its decision to ML_c. ML_c leverages the user request and other data produced by the other system's components (e.g. the user behaviour computed by MS_c and the result of the evaluation performed by AC_c) to decide whether the decision took by AC_c should be changed or the current policy should be updated. The relationship called "Feedback" indicates the possible updating process that ML_c might demand to AC_c and MS_c.

When the system is deployed, it asks ML_c to build the behavioural models from a given training set. The instances contained in the dataset are grouped in different behavioural classes based on the classes of interaction specified as input. The system then derives a set of dynamic policy rules B from the output produced by the machine learning algorithm employed. The set is used to update the original policy π and get a new policy π_{ML}, which will be used by the access control system to perform its tasks.

In the following subsection we describe the inner working of ML_c.

6.3.1 The inner working of ML_c

ML_c is made of three subcomponents, here referred to as CD, UB and IN. The subcomponents serve the purpose of handling the concept drift, training the behavioural models and checking for policy inconsistencies, respectively. ML_c performs different tasks depending on the answer produced by AC_c, after the evaluation of a user request req. If the answer is *deny*, due to the static part of the request, ML_c is asked to verify whether there are policy inconsistencies. Note that if the denial was due to an anomalous user behaviour there would be no policy inconsistencies to search for. In this scenario, ML_c receives the parameters $< req, deny >$ as input (dashed arrow 1). The component resorts to IN (dashed arrow 2) to verify the correctness of AC_c. After this operation, IN notifies its decision to the system

(dashed arrows 3, 4). If the outcome is positive then the system is asked to change the decision and to monitor the user behaviour with respect to the denied resource. If the user behaves similarly to other users that can access to the resource then a new policy rule is generated to address the exception. On the other hand, if req is either accepted or denied due to an anomalous behaviour, a second scenario occurs. This time, ML_c receives the pair $< req, b >$ as input (continuous arrow 1). As shown by the numbered continuous arrows, ML_c asks CD to monitor user behaviours by taking into account b. This operation has two possible outcomes: *1)* a new concept is detected and *2)* there is no new concept. In the first case, CD asks UB to update the behavioural models (arrow 3a) that are affected by the concept drift. After UB completes the updating process, it notifies ML_c (arrow 4a), which in turn notifies ML-AC (arrow 5). In the second case, CD simply notifies that there is no new concept (arrows 3a, 4a).

6.3.2 Notation

In the following we report the symbols used throughout the chapter to denote all the elements that are part of or used by ML-AC.

A user request is referred to as req and the generic output produced by ML-AC as O. We represent a user with u, while a single resource is denoted by r and a set of resources by R. The original policy used by AC_c is referred to as π and any subsequent policy that includes dynamic or behavioural rules, built by means of machine learning, are denoted by π_{ML}. Knowledge about users, resources and available operations is referred to as K_{ac}, while knowledge about the application domain, i.e. the one used to determine the behavioural features of user behaviours, K_d. The classes of interactions that are determined based on K_{ac} and K_d are denoted by C and C_i is the i-th class. With u_i we indicate a user belonging to C_i, instead by using the notation R_i, we are referring to the i-th set of resources which has nothing to do with an existing class C_i. The set of dynamic or behavioural rules that models the behaviours related to C_i is referred to as B_i. A user behaviour is denoted by b and, by adding the subscript i, b_i represents a user behaviour generated by a user u_i. b might represent either a normal or an anomalous behaviour; its nature will be clarified by the context. If an anomalous behaviour is sneaky, e.g. generated by a user u_i and similar to behaviours related to C_j, then it is represented by b_i^j.

6.4 Machine Learning for Policy Refinement

6.4.1 Exploiting machine learning to generate policy rules

In this section we explain in detail how machine learning is employed in order to achieve our goal.

In our initial setting we have a training set containing normal behavioural instances that could belong to different classes of interaction. In this work we assume that only normal behaviours are available because collecting even just a few samples of malicious insider behaviours might be a very difficult task to accomplish [173]. Each of the classes of interaction represents a specific set of normal user behaviours. Given a class C_i, our goal is to recognise if a user behaviour b_i is similar to those related to C_i or not (i.e. it is anomalous). We cannot expect that an anomalous b_i could always match the characteristics of behaviours from a different class C_j. This explains why we cannot consider our problem as a multi-class classification problem, where the labels would be represented by the classes of interaction and our

goal would be to determine if the label of a test sample b_i is C_i or another C_j, with $i \neq j$. To conclude, we want to solve the *One Class Classification* or *OCC* problem [190] for each class of interaction.

Well-known solutions for the OCC problem relies on machine learning algorithms like *Support Vector Machine* or *SVM* [192][191]. We point out that that a machine learning empowered access control systems should be able to explain, to a certain extent, why a user request has been denied or permitted. Therefore, we want to use white box machine learning approaches (e.g. Decision Tree), in contrast to black box approaches (e.g. SVM), because the former provide clues about the reasons behind a classification decision. We chose the *Decision Tree* or *DT* algorithm [151] to pursue our goal because its output might be very useful to generate policy rules. Indeed, the output produced by DT can be converted to a set of rules in the form of $antecedent- > consequent$, as shown in [172], where antecedent is the logical *and* of all the conditions that a behaviour must satisfy in order to be classified as *normal* or *anomalous*, as indicated by the consequent. Such rules are here referred to as *ML rules*, to distinguish them from policy rules. We note that DT is not suitable to solve the OCC problem, because it has been designed to be trained on a dataset containing labelled instances from two or more classes. However, the approach can still be employed to solve OCC by resorting to two possible solutions, i.e. *i)* by generating artificial data to turn OCC into a binary classification problem, e.g. as proposed in [73], or alternatively *ii)* by using a variant of the DT algorithm that can handle OCC (see [62] and [140] for additional details).

Another important aspect to consider is that DT by itself might not be able to build an accurate classifier. Indeed, this algorithm is successfully used as part of the Random Forest (RF) [100][41] approach or Bagging, a machine learning meta algorithm [40]. We then chose to resort to RF for our purposes. A Random Forest consists of an ensemble of DTs. Each DT produces its own set of ML rules, so if we suppose that there are k DTs, RF yields k different sets of ML rules. All these sets are used to classify user behaviours of a specific class of interaction, the one for which RF has been built. When it comes to classify an observation, RF asks its DTs to determine whether the observation is normal or anomalous and subsequently uses a voting algorithm to perform the final classification. Suppose that we have a RF with 3 DTs. Given an observation o to classify, each DT produces its own decision, for example $output(DT_1) = 1$, $output(DT_2) = 0$ and $output(DT_3) = 1$, where 1 and 0 denote the normal and anomalous classes, respectively. At this point, RF decides that o is normal, by means of the voting algorithm.

Despite its accuracy, RF introduces a new problem that concerns the generation of policy rules. The problem lies in the existence of multiple sets of rules and the fact that now a voting algorithm is required to take the final decision, because not all the sets could agree on the nature of a given user behaviour. It would be better to have one single set of ML rules to generate policy rules, therefore we want to merge the sets yielded by RF. There exist a couple of works in literature that address this problem, such as [187] and [10]. We implemented the solution presented in [10] and run it on a RF built from a synthetic dataset that we generated, containing both normal and anomalous behaviours from two classes of interaction. Therefore, we used the original implementation of DT for this analysis. The final set of ML rules obtained by emerging the initial sets was able to classify the user behaviours in a satisfying way.

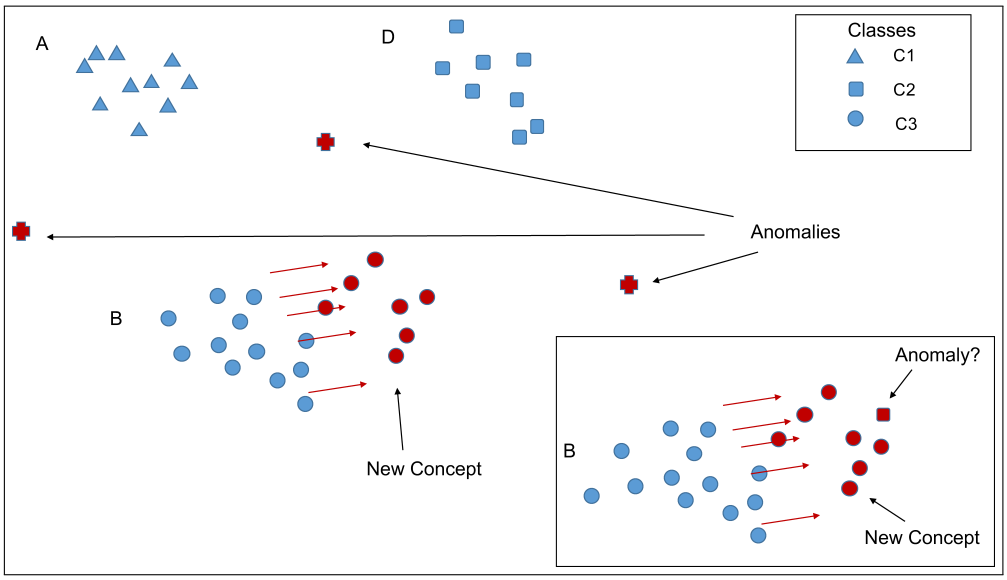


FIGURE 6.3: Scenario where concept drift is detected, along with a few anomalies.

6.4.2 Concept Drift

After the training phase, the system starts monitoring the evolution of user behaviours to detect concept drifts. We don't know whether a new user behaviour is normal or not, in other words the upcoming data are not labelled. Therefore, we need an unsupervised approach to monitor the concept drift and, according to the literature, an approach that might help us is called *Olindda* [183]. Olindda is a clustering-based approach, which has also been used to implement the incremental learning of *BBNAC*. Olindda uses the *k-means* algorithm [99] to detect new concepts. The detection is based on the distance between the new observations/behaviours and the existing concepts (represented by the clusters). The main limitation of Olindda is that it relies on k-means which is suitable only for numerical data. However, if there were categorical features we could replace k-means with the k-modes algorithm [106], which is an extension of the former algorithm.

By employing Olindda, we use two different approaches for modelling user behaviours: we use Random Forest for the task of building policy rules and a clustering-based algorithm to detect concept drift. We need to understand how the results of the latter approach affect the policy rules built by employing the former. To make it clear: when Olindda finds a new concept, which set of policy rules should be updated? The answer to this question lies in what we know about the user behaviours. We know that a user behaviour is associated to a specific class of interaction. Therefore, if a new concept concerns behaviours related to C_i, then we know that ONLY the policy rules that describe the behaviours of C_i have to be updated. Figure 6.3 depicts a scenario where concept drift is observed. In this example there are three types of classes of interaction: C_1, C_2 and C_3. At the beginning there are three clusters of user behaviours, namely A, B and D, containing user behaviours from classes C_1, C_2 and C_3, respectively. At this point, the behaviours are represented by shapes filled with blue. After a certain amount of time, the behaviours from C_3 change to the point that concept drift is detected. The drift is denoted by shapes, representing the class of interaction, filled with red. Because the system knows the origin of the user behaviours, it determines that the set of policy rules that describe C_3 should be

updated. Beside the behaviours involved in the concept drift, a few more instances are observed, those represented by a red cross shape. Such instances are recognised as anomalies by Olindda, because they don't form any cluster and are quite far away from the existing clusters. This is an important aspect to highlight because in order to update the set of policy rules of a class of interaction, we do not want to feed the Random Forest with anomalous behaviours.

It is worth pointing out that it may happen that a new concept mostly concerns behaviours of C_i, but also very few behaviours of other different classes. The box in the bottom right of Figure 6.3 illustrates a scenario where a cluster of new behaviours is made of instances from C_3 and very few instances from C_2. In this case, it might be too risky to consider these few behaviours as normal. Conceptually, these behaviours should be treated as anomalies, because they are in a cluster the majority of whose members are part of a different class of interaction. The system should notify a domain expert about these few behaviours so that appropriate decisions could be taken.

6.4.3 Automated exception handling

The approach we chose to detect unforeseen accesses to be granted is inspired by the work presented in [25]. The work proposes the usage of *association rules* to predict policy misconfigurations from user activity logs that describe the resources that are accessed by the users. By policy misconfigurations the authors mean the absence of one or more rules in a policy that prevent one or more users from accessing to specific resources. In other words, the access control system answer to a valid user request with *deny* instead of *permit*.

In this context, each accessed resource is represented as an attribute and a set of resources used by a user form a record. The idea behind the usage of association rules is to find significant subsets of attributes in the records describing user activities. From these subsets a set of rules is derived, in the form of $premise \Rightarrow conclusion$, which state that if the attributes in the premise are present in a record then the attribute in the conclusion should also be present. In order to solve the problem the authors assume that knowing the past user resource usage patterns can provide useful information to determine if a user is likely to access to a resource even if the access control system denies it.

To make the approach clear we provide an example which is based on the policy shown in Listing 6.1. Suppose that Alice, a junior manager, wants to access to a resource r that is needed to complete certain tasks, but her request is denied, due to a negligence or a configuration error. The resource is also used by other users, including the senior manager that is assisted by Alice. Moreover, some of these users utilise a set of resources R_i that are related to r, in that when the users access to resources that belongs to R_i, they typically access to r. Some of the resources in R_i are also used by Alice and it happens that these resources are related to r. The work proposed in [25] is able to infer that the authorisation should be granted to Alice, based on the past activities of all the other users that are authorised to access to R_i. The main problem of the approach is that it tends to generate a considerable number of wrong predictions that do not make sense. Consider the previous example. Suppose now that the users that can also access to r perform activities that have nothing to do with those conducted by Alice. Alice might be involved only in activities concerning Project A, while the other users works on different projects. In this case, it makes no sense to authorise Alice to access to r, because the resource is used in different contexts. However, the approach proposed in [25] would suggest to let

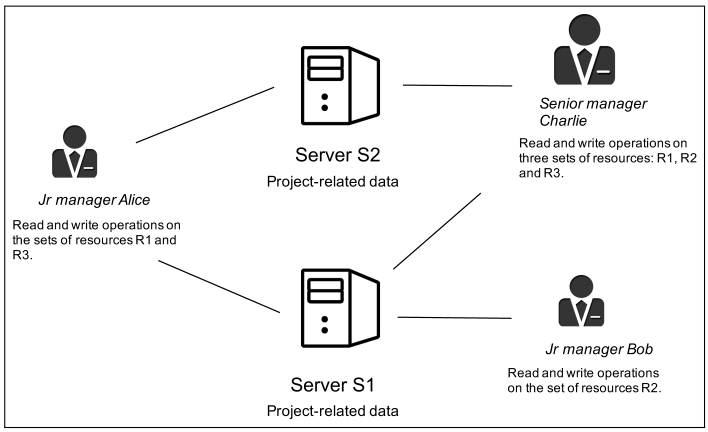


FIGURE 6.4: Interactions between users and resources.

Alice use r by looking at the activities performed by the other users. We believe that by combining the mentioned approach with contextual knowledge we can obtain a useful tool to detect inconsistencies. We want to exploit knowledge about users and resources to identify significant resource access patterns based on which we take decisions. In the example, knowing that the other users are working in a completely different context, ML-AC would not change the decision taken by AC_c, therefore Alice would not be able to access to r.

The assumption that constitutes the basis for the chosen approach is likely to not hold in every application domain. In this case, the solution might not work very well. Therefore, ML-AC should feature a modular architecture, so that it is easy to integrate new solutions that work better under different assumptions.

Using Association rules

This section explains in detail how we use Association rules. First of all, ML-AC runs an association rules algorithm, like *Apriori*, on a given user activity log. Specifically, the log contains the activities of all users, where each activity describes a set of resources accessed by an individual. An activity is mapped to a record, which is represented as a set of attributes $(a_1, a_2, ..., a_m)$, where m is the record length. Afterwards, a set of rules Θ is derived, based on user-specified confidence and support values.

When ML-AC receives a user request req such that AC_c answers with *deny*, the system communicates the outcome to ML_c. ML_c determines the type of user who is requesting the denied resource based on criteria determined by a domain expert. The criteria should suggest what is the level of privilege assigned to the user and who are users with similar characteristics (i.e. compatible users). At this point, the system searches for a subset of rules $\Theta_R \in \Theta$ such that the conclusion of the rules is the denied resource R. If $\Theta_R \neq \emptyset$ then ML-AC checks whether there exists a rule $\theta \in \Theta_R$ so that *i)* its premise is satisfied by the requesting user and *ii)* both its premise and conclusion are satisfied by another compatible user. If the outcome of the mentioned operation is positive then ML-AC changes the answer produced by AC_c from *deny* to *permit*.

6.5 Case study

We suggest as an application case study the management of two software projects, *A* and *B*, from an access control point of view. For the sake of simplicity, only aspects that are needed to explain how ML-AC works will be taken into account.

Figure 6.4 depicts the IT figures and the resources involved in our case study and how they interact. The former group includes *junior and senior managers*, while the latter consist of two servers, i.e. S_1 and S_2. The servers contain project-related documents.

The following sections provide the description of the organisation needs, part of which are described in the portion of policy that we introduced in Section 6.2.1, and of the case study scenarios.

6.5.1 Policy

In this section we highlight the interactions that are useful to present the application of ML-AC in the case study.

There are two *junior managers*, Alice and Bob, who are assisting Charlie on Project *A* and *B*, respectively. Similarly to Charlie, both Alice and Bob are authorised to access to the S_1 server, but Alice can only read and/or write a set of resources R_1, while Bob can perform the same operations only on the set of resources R_2. Charlie and Alice can also access to the set of resource R_3, stored in the S_2 server.

6.5.2 Case study description

We propose four scenarios to show the capabilities of the system, namely i) sensitive data theft, ii) access wrongly denied, iii) access correctly denied and iiii) behaviour evolution. The first scenario describes an attempt of Alice to steal sensitive data from the server S_1, in a fairly brief period of time. Such attempt leads to the generation of an abnormal amount of downloaded data. In the second scenario Alice tries to access to a document that is important to complete tasks related to her work, without success. The request is denied due to a negligence from the policy administrator. ML_c overwrites the output produced by AC_c, because the junior manager shows resource usage patterns similar to those of other compatible users that can access the resource. The third scenario sees Alice trying to access to a document that can only be accessed by Charlie and Bob. We show two possible scenarios where Alice's request cannot be allowed. At last, the fourth scenario shows what happens when new emerging user behaviours are detected, in other words the concept drift. Specifically, we show that in the moment the system realises that the behaviour of Alice and other colleagues has changed, it starts a process whose goal is to update the dynamic policy rules.

6.5.3 ML-AC in action

Before proceeding we recall the features that are used to represent user behaviours: the average number of bytes read (or *BytesRead*); the average number of reads performed (or *NumberOfReads*); the average number of bytes written (or *BytesWritten*) and the average number of writes performed (or *NumberOfWrites*). The features are computed so that they describe how a user behaves in a time window of 10 minutes.

In the first scenario we observe the action taken by Alice who wants to steal sensitive data from the S_1 server. She is part of a class of interaction C_1 which is

defined as the triple <(*Junior manager, Department 1, Project A*); (*read, write*); (R_1)>. The class C_1 is characterised by: *1)* users who are junior managers that are assigned to *Department 1* and work on *Project A*; *2)* read and write operations and *3)* R_1, a set of resources stored in the S_1 server. The listing 6.1 introduced at the beginning of this chapter describes a portion of the original ABAC policy π used to check user requests related to C_1. As explained in Section 6.3, after the deployment of ML-AC ML_c builds a behavioural model for each class of interaction and from each $i-th$ model a set of dynamic policy rules B_i is derived. Therefore, ML-AC builds B_1 that is used (along with other sets) to modify π to get a new policy π_{ML}. The current policy captures dynamic aspects concerning user behaviours, as can be seen in Listing 6.2. The attributes *BytesRead*, *NumberOfReads*, *BytesWritten* and *NumberOfWrites* have been introduced, which are used to describe how the users from C_1 are expected to behave when they use any resource $r \in R_1$. For the sake of simplicity, the rules reported are very simple. In reality, many more rules might be needed in order to properly describe user behaviours from a class of interaction. Going back to the main actor of this scenario, at some point Alice decides to act in a short time window to steal as many documents as possible from the server. We assume that the actor issues many normal requests in a short timeframe. That means that at the beginning Alice is allowed to access to the resource because her behaviour is still normal. However, after a couple of requests the behaviour starts deviating too much from the expected one. The deviation is detected by ML-AC, therefore it will deny the access.

In the second scenario we focus on the senior and junior managers. This time we analyse what happens when Alice tries to access to a document called *doc0.docx* that she needs in order to complete tasks related to her work. Alice is working on *Project A*, as well as Charlie, the senior manager. Due to a negligence from the policy administrator, she cannot access to *doc0.docx*. Therefore, when ML-AC receives a request from Alice for *doc0.docx*, AC_c denies the access. Before ML-AC produces the output, ML_c analyses the user request to determine if AC_c is correct. ML_c determines the type of user, i.e. a junior manager that is working on *Project A* from *Department 1*. Afterwards, the component searches for other types of users that are compatible with Alice that can access to *doc0.docx*. ML_c finds out that Charlie is a compatible user. The next step consists of verifying if there exist a set of rules where the conclusion corresponds to the requested resource. ML_c finds the following patterns: $\{doc1.xls, doc3.docx\} \rightarrow \{doc0.docx\}$ and $\{doc2.xls\} \rightarrow \{doc0.docx\}$. It then checks Charlie's resource usage patterns. The outcome suggests that when users like Charlie access to *doc0.docx*, they exhibit the following pattern: $\{doc1.xls, doc3.docx\} \rightarrow \{doc0.docx\}$. The patterns are presented as rules built by means of association rule mining (see Section 6.4.3 for additional details). According to the rules, *doc0.docx* is typically used along with the resources *doc1.xls*, *doc3.docx* and *doc2.docx*. To conclude ML_c checks whether there exist a Alice's usage pattern that matches Charlie's pattern. In this case the answer is positive, therefore ML_c asks ML-AC to overwrite the answer of AC_c and let Alice access to the desired resource.

The third scenario sees Alice trying to access to a document *docb.docx* with the intent of damaging the reputation of Bob. The resource is typically used by Bob and Charlie for tasks related to *Project B* and Alice should not access to it, indeed when ML-AC receives her request, AC_c answers with a *deny*. ML_c performs all the steps previously described, but this time Alice will not be able to get the authorisation. Two possible possible outcomes could be observed. In one case, there could exist another compatible type of users that can access to *docb.docx*, but use resources that are different from those used by Alice, due to their job role. ML_c will not be able to

find matching patterns, therefore it will not change the answer of AC_c. Alternatively, there exist no compatible user, so ML_c immediately concludes that Alice should not access to the resource.

In the last scenario we observe what happens when concept drift is detected. Listing 6.6 shows a portion of π_{ML} that describes the behaviour of users that access to resources of type R_3, at time t_1. In this time instant the users are struggling to meet a deadline. To achieve their goal they need to work primarily on the set of resources R_1, therefore they are performing a few reads and writes on R_3. After the deadline, they start working at a higher pace with resources in R_3, so their behaviour starts to change. ML-AC receives a sequence of user requests related to user behaviours that are significantly different from previous ones. ML_c recognises a new concept and at time t_2 it decides to update the behavioural model that describes the behaviours related to the usage of R_3. New dynamic policy rules are derived and π_{ML} is updated accordingly. Listing 6.7 shows the new portion of π_{ML}.

```
Policy set policy1 {permit-overrides
      target: equal ("R 3", resource/type) policies:
Rule rule1(permit target:
            equal ("Read", action/action-ID)
            && equal ("Junior manager", subject/role)
            && equal ("Department 1", subject/department)
            && equal ("Project A", subject/project)
            && greater-than (feature/BytesRead, 921.8)
            && greater-than (feature/NumberOfReads, 8)
            && less-than (feature/BytesWritten, 1183.2)
            && less-than (feature/NumberOfWrites, 12)
      )
}
```

LISTING 6.6: A portion of π_{ML} that describes the behaviour of junior managers at time t_1.

```
Policy set policy1 {permit-overrides
      target: equal ("R 3", resource/type) policies:
Rule rule1(permit target:
            equal ("Read", action/action-ID)
            && equal ("Junior manager", subject/role)
            && equal ("Department 1", subject/department)
            && equal ("Project A", subject/project)
            && greater-than (feature/BytesRead, 2458.6)
            && greater-than (feature/NumberOfReads, 35)
            && less-than (feature/BytesWritten, 3239.1)
            && less-than (feature/NumberOfWrites, 51)
      )
}
```

LISTING 6.7: A portion of π_{ML} that describes the behaviour of junior managers at time t_2.

6.6 Classes of interaction evaluation

In the previous sections, we showed how to exploit user behaviours to improve access control. As discussed in Section 6.1, another work that perform access control based on user behaviours is *BBNAC*. *BBNAC* replaces policy rules with behavioural models, so a user request is evaluated only as a function of his behaviour. However,

this approach suffers from a coarse-grained modelling. *BBNAC* builds a distinct behavioural model for each network protocol (e.g. HTTP, FTP, etc), in order to separate different types of behaviour. However, it does not properly address the types of anomalies that we discussed in Section 6.2. Suppose that there are two subpopulations of HTTP behaviours, here referred to as A and B, and that *BBNAC* represents each of them with a cluster. If a user from A generated a malicious behaviour that is very similar to those observed in B, *BBNAC* would classify it as normal because there exists a cluster in which it belongs to.

We set up an experiment to demonstrate that combining a machine learning algorithm with domain knowledge and knowledge encoded into a policy has significant benefits. Thus, we compare ML-AC with *BBNAC* , which is the closest approach to ours from the literature, and show that we can better classify anomalous behaviours. We also perform a comparison with a variant of ML-AC that ignores contextual knowledge , to better highlight the role of contextual knowledge. We note that in [160] an experiment to evaluate the impact of the separation of different types of user behaviours on a variety of ML algorithms was presented, therefore we will not propose an experiment with the same goal.

We generated a synthetic dataset D for confirmatory purposes [92], containing user behaviours that belong to two different classes of interactions. Specifically, we generated two sets of random points defined by Cartesian coordinates. The points of each set are uniformly distributed within a 3-dimensional hypersphere of radius r. The hyperspheres were generated in a way that there is no intersection between them. Each hypersphere represents a distinct class of interaction, i.e. a population of legitimate behaviours. Like this we simulate an OCC setting, where only normal user behaviours are available. In order to turn the OCC problem into a binary classification problem, we generated a cloud of anomalous points around each hypersphere. The cloud consists of a set of random points generated on the surface of an hypersphere which shares its centre with the hypersphere containing normal behaviours and has a radius slightly greater.

Depending on the knowledge used by the machine learning approach, different training and test sets can be built. We have three different settings for this experiment. The first setting is used for ML-AC, when it takes into account the existence of the classes of interaction. We have two training sets, one for each class of interaction. The training set of the class C_i contains both normal and anomalous behaviours generated by users from C_i. The test set of the class C_i not only contains unseen normal and anomalous behaviours related to C_i, but also behaviours that are indistinguishable from those generated by users from C_j. With this addition we can simulate sneaky anomalies. The second setting of the experiment concerns ML-AC when it ignores the classes of interaction. The training set is made of normal and anomalous behaviours from both the classes. The test set contains unseen normal and anomalous behaviours and also sneaky anomalies from both the classes. We expect ML-AC to struggle in recognising the true nature of such behaviours, in this setting. At last, the third setting is very similar to the second one, in that the test set is the same while the training set contains only normal behaviours. We computed the ROC curve for each experimental setting, in order to evaluate the accuracy of the approaches. Figure 6.5 shows the results of the experiment. As can be seen, ML-AC achieves the best performances, hence the role of additional contextual knowledge about the classes of interaction is of paramount importance. The curve associated to ML-AC$_{nok}$ is significantly worse. By ignoring the additinal knowledge, the AUC drops from circa 99% to circa 87%. The performances of BBNAC are comparable to those of ML-AC$_{nok}$, its AUC is about 85%. More in details, both BBNAC and

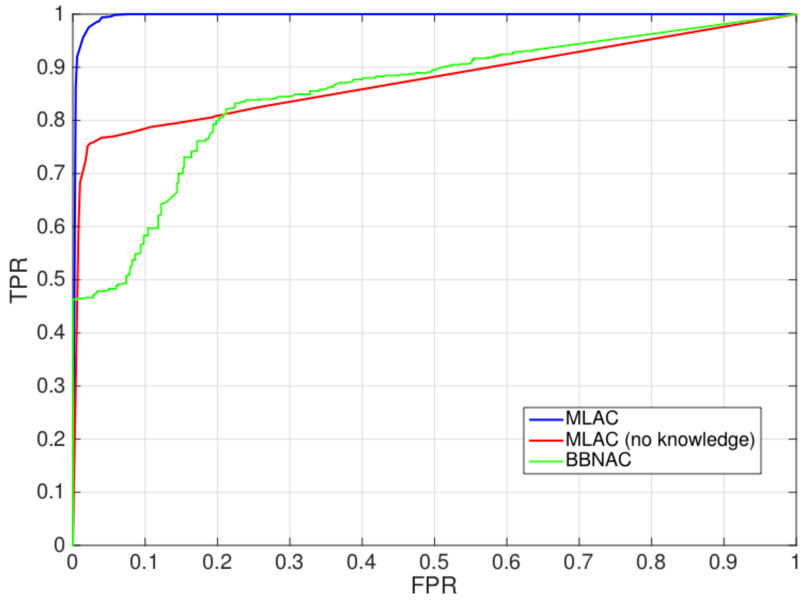


FIGURE 6.5: Comparison among ML-AC, *BBNAC* and ML-AC$_{nok}$

ML-AC$_{nok}$ have no problem in classifying the normal instances, their lower accuracy is mainly due to the fact that they consider the majority of the sneaky anomalies as normal behaviours, because they are indistinguishable from the normal samples.

6.7 Summary

In this chapter we described a machine learning solution thought to refine and update access policies in a dynamic fashion, with the goal of preventing insider threats and enabling automated policy administration. The approach, called MLAC, generates new access control rules based on runtime user behaviours. These rules are used to detect legitimate users with malicious intentions. In addition, MLAC exploits association rule mining to detect policy inconsistencies.

Chapter 7

Using Virtual Environments for the Assessment of Cybersecurity Issues in IoT Scenarios

This chapter presents the work conducted on the Internet of Things or IoT.

During the last few years, Internet applications are still increasing and, in particular, an outstanding number of highly heterogeneous networked objects (things), many of which characterized by small size and low power consumption [148] are becoming part of Internet, e.g. implantable medical devices, smart thermostats, smart meters, or any object that has the ability to transfer data over a network. This trend has been widely recognized as the next main step in the evolution of Internet which is commonly referred to as the Internet of Things (IoT). IoT is showing the potential for impacting several domains, ranging from personal to enterprise environments [18]. Examples of domains and possible applications include, but are not limited to, smart cities, for lowering energy costs and reducing pollution, and smart homes, for which energy companies are building systems to increase energy savings and safety.

Despite the goals of IoT applications are directed to improve most aspects of both business and common people's life, such emerging technology has become an attractive target for cybercriminals. The more are the Internet connected devices the more are the potential attack vectors and the vulnerabilities that malicious entities may exploit. Estimates on the number of devices that will be connected to the Internet by 2020 range from 20.8 billion [86], to 30 billion [194] devices. As reported by [8], by 2016 cybersecurity risks have received little attention. Recently, positive signs have been observed, starting from the growing attention received from governments [65] to the announcement of bug bounty programs for the discovery of vulnerabilities.

In spite of the greater degree of attention, there are still manufacturers that do not properly take into account the security aspects of their products: the services provided by the devices get almost all the attention, while security aspects receives little consideration [114]. This means that the security state of smart devices is low, it is then easier to observe more vulnerabilities and, as a consequence, there can be more attack vectors to be exploited by cyber criminals. An important factor behind this problem is the presence of manufacturers that lack prior experience with networked devices: in an attempt to place into the market their devices and get the newest and attractive functions at the lowest cost, as quickly as possible, they end up neglecting the design and implementation of security features for hardware and software.

It is of utmost importance giving to security a high priority during the development process of IoT, otherwise, in the near future, the number of security risks for consumers and businesses will increase exponentially, leading to disastrous situations for both sides. Therefore, security should not be an artefact added at the

end of the development, but it must be an integral part of the entire process. Consequently, the devices placed in the markets, should be equipped with built-in security mechanisms and ensure greater protection for their users.

To address the security vulnerabilities of IoT devices created so far, researchers are focusing on the evaluation of security properties [159]. The goal of this analysis is to identify and understand the security issues of currently deployed devices and help manufacturers to solve the detected problems, by providing them with guidelines and recommendations for improving the security of future software updates and/or version of the devices. Towards this objective, computer simulation techniques along with novel cloud based virtualization platforms represent a very good combination for achieving suitable cybersecurity analysis and assessment platforms. Virtual environments are systems in which realistic scenarios can be reproduced, by exploiting computer and network virtualization technologies and agent-based simulation [49, 51]. They find applications in many domains including military, medical, educational and recently also in cybersecurity [82].

We show how virtual environments can be a valuable tool to assess security properties and discover vulnerabilities of IoT devices, in realistic scenarios. Specifically, the SMALLWORLD platform is proposed for the development of intelligent virtual environments in which the agent paradigm is used to simulate malicious and legal behaviours, both of machines and human beings. SMALLWORLD has being developed to be scalable by design. It introduces an abstraction layer and a set of API which make it able to run on different hypervisor technologies ranging from single machine solution (e.g. VirtualBox) to state of art cloud solution.

With this contribution we propose a solution that on one hand helps to mitigate the presence of security holes in IoT devices to prevent anomalous actions from taking advantage of them for malicious purposes. On the other hand, the solution allows the collection of data that can be analysed to evaluate the security properties of the devices of interest.

The rest of the chapter is organized as follows. Section 7.1 gives an overview of the related work. Section 7.2 discusses the main security issues affecting IoT technologies and devices as they are currently developed, implemented and deployed. Section 7.3 describes the use of virtual environments as a security analysis assessment tool. Section 7.4 presents a case study involving smart home applications. Finally, Section 7.5 summarises the work.

7.1 Related work

A model-based security toolkit, SecKit [159], has been proposed to enable the protection of user data by supporting specification and efficient evaluation of security policies. SecKit is integrated into a generic management framework for IoT devices. It has been designed to support the modeling of IoT systems and to specify, in an integrated way, security requirements, usage control policies, threat scenarios and trust relationships. These issues are addressed by means of meta models and a policy rule language. In particular, the adoption of trust models enables the specification of trust relationships, of various types, by which governing the trust relationships in the IoT interactions. The use of SecKit was experimented in [159] under two scenarios, regarding smart home and city.

In [149], the authors present ASTo (Apparatus Software Tool), a software tool designed for analysing security properties of IoT systems. The analysis can be performed during the design and the implementation phases of the system. The tool

uses a domain specific modelling language to visualise IoT systems in terms of hardware, software and social concepts, along with security concepts. By analysing the attributes and relationships of the constructs provided by the language, security issues can be identified.

A framework for modeling and assessing security in the IoT is described in [87]. Specifically, the objectives of the framework are to graphically represent all possible attack paths in an IoT network, whose configuration is provided as input by a security decision maker, in order to evaluate the effectiveness of possible defence strategies. The use of the framework has been experimented in the domain of pervasive healthcare monitoring and of environment monitoring. However, its main limitation is due to the fact that it requires a sensor networks of identical nodes which is very unlikely to occur in real IoT settings. In addition it is not able to take into account the mobility of devices.

Another general security assessment framework for IoT services is discussed in [167]. The proposed approach uses integrated fuzzy multi-criteria decision-making methods. It exploits a combination of a fuzzy analytic network process (ANP) and of the fuzzy decision-making trial and evaluation laboratory (DEMATEL). The former is used to assign a weight to each IoT security requirement, the latter is employed to derive cause-and-effect interrelationships between the security criteria. The framework aims at handling both qualitative and quantitative security criteria.

Security issues regarding the use of IoT in the field of eHealth applications are considered in the work described in [139] where a framework for the assessment of context-aware adaptive security solutions is applied in this context. A set of IoT-eHealth scenarios is provided and it is considered for the evaluation of the approach. Further, the framework employs linear and logarithmic approaches to assess and quantify the security and QoS requirements of the applications, in an adaptive security system. The evaluation methodology is based on a comparison between results from laboratory experiments and simulations and the assessment by human observers. The work presented in [139] is complemented by that discussed in [6], where, in addition to QoS and security requirements, user preferences and device capabilities are also taken into account.

A metric-based approach to assess the security level of IoT connected Critical Infrastructures (CI) is proposed in [89]. The authors introduce a set of suitable security metrics on the basis of which the satisfaction of security requirements is checked. The metrics are employed to define Service Level Agreements (SLAs) in which the requirements and the penalties that must be applied in case of violations are defined. The approach is evaluated in the context of a financial infrastructure, however, the approach is generic and can also be applied to other CIs. A discussion on the specific security issues related to IoT applications for the Smart Grid is reported in [26].

While the above described approaches have their merits in the view of design of secure IoT applications, they are mostly based on theoretical models from which evaluation framework are derived and, as a consequence, they tend to overlook some practical details which may hide serious security holes that can be lately discovered only when IoT systems are put into operation. In order to fill this gap, the availability of a platform allowing to reproduce in a realistic way (part of) an IoT infrastructure accounting for low level operation details (e.g. operating system and employed library versions, specific hardware and software components, network topology, firewall rules, etc.) is of critical importance. To the best of our knowledge, this work is the first proposing the combined use of virtual environments, agent-based simulation and real devices in order to allow accurate evaluation and assessment of realistic IoT deployment scenarios which may involve complex networking

infrastructures.

7.2 Security concerns

IoT has brought interesting opportunities both for consumers and businesses, however it came together with vast repertoire of new security challenges. IoT technologies are *embedded* into and extend the Internet ecosystem and, as a consequence, they inherit all the Internet related security problems and pose new specific issues. Because of the pervasive nature of IoT devices and applications, these security problems are of a greater importance and, in some cases, tend even to become critical. To cite one example, a group of researchers, in 2008, showed how it was possible to extract personal information from a pacemaker or even to threat the life of a patient by altering the behaviour of the device [96].

Similarly to the Internet, the IoT can be subject to a high number of threats, such as attacks that target diverse communication channels, physical threats, denial of service, identity fabrication, and others [23]. Unlike the Internet, in the IoT the attack surface increases exponentially given the high number of interconnected devices. The current state of IoT is also characterized by the absence of standards and the extremely heterogeneous nature of the devices, in the hardware, software and adopted communication protocols (Wi-Fi, Z-Wave and ZigBee, to name a few). All these conditions introduce considerable complexity into the design process of the general security solutions. Moreover, in a typical Internet scenario the connected resources, e.g. desktop computers, have enough computing power to run software tools, e.g. antivirus, that can protect them from some threat sources. In the IoT instead, the devices have limited resources and because of this the use of existing technologies such as antivirus is often impracticable.

Having said that, it is clear how important it is to pay special attention to the security topic in the IoT. Unfortunately, it would seem that the IoT is retracing the same steps of the Internet in its infancy. Decades ago, when the Internet was going through its early stages of evolution, those who devoted themselves to its design and development were focused on technical issues with the aim of being able to transfer information quickly and reliably. These people were shortsighted about information security, they primarily took into account military threats, but they failed to understand that the same Internet users a day might become threats. This led to a situation in which it was necessary to introduce ex post information security solutions in response to a very high number, with variable severity, of threats. It is desirable to undertake a change of direction, i.e. by stopping to neglect the security aspect and by investing resources to design the devices taking into account the possible security issues that may afflict them in the future. In this way, devices with built-in security will be introduced in the market, offering a better protection to the customers. IoT only recently has received a significant amount of attention from a security perspective. In a study conducted by HP [8], it was found that the security topic in the IoT has had a very weak presence in industrial and academic conferences in 2014 and 2015, with respect to other application domains. Nowadays, IoT has attracted attention from governments [65] and there are bug bounty programs thought to discover vulnerabilities in IoT products. Nonetheless, there are still manufacturers that are devoting much of their attention to the development of capabilities and technologies with the objective of achieving long-awaited services and get a rapid spread in the IoT market [114]. Therefore, devices with a security holes get sold and put in operation. An example of security hole that has caused significant problems is the

availability of the telnet service. Malwares like Mirai [14] and Bashlite [113] have taken advantage of the telnet protocol to build botnets made of compromised IoT devices with the ultimate goal of launching DDoS attacks [71].

The following subsections provide more information on the security landscape of IoT. Specifically, they describe what are the main threats for the IoT, what are the motivations that push malicious users to act, what types of attack patterns can be observed and some examples of exploits.

7.2.1 Threat Sources

IoT devices manage a huge quantity of information, related e.g. to lifestyle habits of a consumer, and they are capillary distributed in every industry [28]. This aspect is the main reason why the IoT security is threatened. Criminals, government entities, and hacktivists are just few examples of actors who harbour interests with respect to these data. For example, a group of criminals might be interested in stealing sensitive information by hacking specific devices. In this scenario it becomes potentially easier also to observe cases where a person, for personal reasons, may disturb the daily life of another person, by altering the normal functioning of the devices installed in the victim's home.

Three main categories of malicious entities threatening IoT can be identified [17]: *i)* external attackers, *ii)* malicious users and *iii)* bad manufacturers.

An *external attacker* is an entity that does not have permission to access a system or a device. He usually remotely target a device (or set of devices) by exploiting its vulnerabilities and he can have various goals, e.g. stealing sensitive data, causing malfunctions or financial damages.

A *malicious user* is identifiable as the owner of a device from which he wants to extract data relating to secrets of a manufacturer, or gain access to features not accessible to the user. One of its objectives could be to sell secrets to a third party, e.g. in the case of a former employee of a company, driven by resentment.

As malicious users may be interested in obtaining sensitive information from a manufacturer of a certain device also the opposite situation could happen. A *bad manufacturer* might be interested in gathering information about its general users or about a specific user's habits. To achieve this result it could deliberately introduce security holes, by means of which it is possible to gain access to user data, violating his privacy. A manufacturer could also be interested in seeking information on other IoT devices or it might even try to attack other devices, produced by competitor firms, in order to damage their reputation.

7.2.2 IoT exploit scenarios

This section reports two cases of IoT device exploits in order to make clear the impact that the presence of vulnerabilities in smart devices can have on people's lives. During the 2015 edition of the Black Hat USA security conference, Miller and Valasek showed how they were able to compromise a smart car, specifically a Chrysler's Jeep [152]. The two researchers explained that there were two ways to perpetrate the attack. In one case, the victim must had to be subscribed to the wireless connection service from the manufacturer. They found out how the Wi-Fi password is generated, i.e. based on the default system time plus a few seconds due to the boot procedure of the head unit. The date corresponded to `January 01 2013 00.00 GMT`, and in the specific case study to `00.00.32 GMT`. The number of combinations

to be generated was small, therefore, little effort were required to guess the password. Once a connection was established with the Jeep's head unit, it was possible to find a way to hack the multimedia computer, which runs on a Linux operating system. They managed to take control of the head unit of the system by exploiting some pretty guessable flaws in the software. Of course, not all consumers can be interested in signing the service offered by the manufacturer. In the second case study, they shown that it was possible to obtain control of the system, leveraging the connection that all head units had with the Sprint cellular network. More details can be found in the article published by Miller and Valasek.

Any type of device may be part of the IoT, including dolls. In 2013 Mattel has put on the market *Hello Barbie*, a doll which uses Wi-Fi to transmit what children say to it to remote servers that process the speeches and build suitable replies. Researchers showed that the doll had few insecurities. Studies conducted on Android and iOS applications associated with the doll, revealed the presence of serious defects by which an attacker is able to eavesdrop on communications between the cloud server and the doll [181]. Furthermore it was showed that the application will automatically connect to any Wi-Fi network whose name includes "Barbie". These flaws were exploited to gain access to system information, Wi-Fi network names, internal MAC addresses, account IDs and MP3 files. Furthermore, these data can be used to find someone's house and access personal information.

7.2.3 Attack vectors/models

In the world of IoT, old and new attack patterns arise. When IoT will have reached full maturity, smart devices will be everywhere, e.g. in our homes and offices. This will allow an attacker to be able to get physical access to a device, i.e. the highest level of access. Although it may seem hard to believe, physical access will be a plausible attack vector. Just think of a guy who still has access to the home of a former girlfriend, he will have access to the devices and try to reconfigure them to spy on the movements of the victim. The attacker could exploit the physical access to capture a device and extract the information contained in it or alter its configuration, an attack pattern called node capture [168]. It would therefore be possible to reset the device in an attempt to restore its original settings, or install a custom Secure Socket Layer certificate for directing traffic to a server under his control. It is also conceivable that an attacker compromises a device in his possession and resells it to spy on other people. This new attack vector also allows to conduct a Denial of Service attack. Indeed, if the attacker is able to access a device, it might make it unusable, by destroying it.

The above mentioned Denial of Service (DoS) attack is another serious threat to IoT. A DoS attack is defined as any event that diminishes or eliminates a network capacity to perform its expected function, degrading the quality of the services offered to its users [16]. These attacks can be initialized from remote places with mere commands, combined with advanced tools. Distributed DoS attacks may also be performed, which are more effective in exhausting the networks' resources. In the IoT one of the primary objectives for this threat is the wireless communication infrastructure. By using attacks like Jamming [213], which is a special case of DoS attacks which interferes with the radio frequencies used by sensor nodes, an attacker may prevent communications between smart devices, making it impossible for exchanging information, a vital aspect in the IoT.

Another longtime threat reoccurring in the IoT is the malware spread. The first malware in the Internet of things was discovered by Symantec in 2013, and was

named `Linux.Darlloz` [189]. Malwares are a very powerful means to compromise a device. It can be exploited to reach another device that contains the data of interest. Unfortunately the limited resources available to the devices, make it hard to deal with this threat. It is not possible to use tools like antivirus, in order to recognize malwares in real time, because they would require an unsustainable strain on the device.

In the following we report a number of works that propose solutions to various problems. To cite one example, the work published in [143] copes with the presence of loopholes in device security and data integrity, by proposing an access control and authentication mechanism. The method requires that a user has to authenticate in order to access a device and asks for permission from a Registration Authority. The Authority, in turn, sends the user a challenge, if the answer is positive, then the user is authenticated and can access the device. Unfortunately, the proposed solution cannot prevent systems from being very vulnerable to Man in the Middle and Eavesdropping attacks. Other solutions are discussed in [134]. For each proposal, the authors examine what are the issues that are addressed and the corresponding limitations.

7.3 Virtual environments

Modelling and simulation techniques are essential engineering tools allowing human beings to study, analyse, understand and predict the behaviour of often complex real phenomena. It is of critical importance the ability to achieve suitable mathematical models which: are accurate enough to describe the entities under investigation, are computer executable and abstract away from superfluous details. By composing models of different entities it is possible to design new complex systems which, once implemented, will interact with the real world. Simulation allows to take important decisions at design time, e.g. on the basis of the results of what-if analysis coming from the playing of different scenarios.

In the literature, two are the main categories of simulation applications that have been identified [81]: *analytic simulations* and *virtual environments*. The first includes *traditional* applications of computer simulations whose main goal is to achieve quantitative evaluations about what is being simulated and which during the execution include little or no interaction with the real world (human beings and/or physical devices). Analytic simulations are run *as fast as possible* and must supply reproducible results. Virtual Environments (VE), are systems able to simulate highly realistic environments with which people, physical devices and other systems may interact. As a consequence, a fundamental requirement for a VE is that its state must evolve at the same pace as it would in the real world so that external entities can perceive realistic feedback to their interactions with the VE. Virtual Environments are used in many areas including: military, medical, educational, emergency management and gaming. During last few years VE started to be used in the field of cybersecurity [82, 175]

VEs can reuse most of the techniques that have first been devised for analytic simulations, e.g. discrete-event [81] and agent-based [49] simulations, by synchronizing the simulated time with the wallclock time.

Agents are entities able to reproduce complex human or system behaviour into a scenario (real or simulated). They can operate without human interaction and perceive the environment around them. The behaviour of the agents can be defined by a finite state machine [50], a Petri net or can be established by equipping the agent

whit artificial intelligence algorithms [210]. Agents generally are able to cooperate and coordinate with each other in order to achieve a common goal. As will be shown in the following sections, agents are used to animate IoT scenarios by playing the roles of legal/malicious entities.

In the last few years, high-performance hardware virtualization [97, 7] technologies allowed to realize complex computer networks whose nodes are virtual machines, each executing its own OS, applications and services. These technologies in turn allowed the development and diffusion of software defined networking [132] (SDN) which is currently exploited by cloud services vendors like Amazon and Google. All of this has been possible because software and protocols implementations are mathematical objects [90] and then they can be used as models of themselves.

The combined use of hardware virtualization, agent-based simulation and real devices (e.g. IoT devices) allows the realization of VEs that are suitable for the assessment of complex infrastructures in the field of Information and Communication Technology (ICT) and, in particular, cybersecurity related aspects. Entire ICT infrastructures or relevant parts can be deployed in such a VE along with agents running on suitable simulation engines deployed on some VMs.

In this area some critical infrastructures like banking systems now have a high degree of dependence on ICT. This bound carries with it substantial advantages, e.g. automation of processes, but, on the other hand, introduces problems including security vulnerabilities. These vulnerabilities arise for several reasons, such as poor code quality. Unfortunately, many of these vulnerabilities are difficult to find, due to the systems complexity.

Different approaches have been studied in order to identify the vulnerabilities, including penetration testing. For example, a malware could be injected in a set of interconnected nodes, in order to study its propagation within the network and possible mutations. Unfortunately, this approach presents considerable risks, due to the unpredictability of the behaviour of the test, which could lead to inconsistent and perhaps irreversible states of the system. A similar problem can be observed when studying the resilience of critical systems, another important research topic [88].

Researchers can rely on emulation for such analysis, especially approaches based on Emulab software [179]. Emulab is a network testbed, in which a great variety of experimental environments can be reproduced, that enable the development, testing and assessment of complex systems. In this way, the exposure of the real system to high loads and extreme conditions is avoided.

However emulators suffer from several shortcomings. The agent paradigm is not applicable, therefore the use of these software components must be reproduced by means of human intervention. It is not possible to create a distributed environment, unless the installation of a number of emulators equal to the number of nodes of the system to be analysed is performed. Finally, one of the major difficulties in the use of emulators is due to the need of obtaining specific software or hardware components for the system to study.

For the reasons exposed above, emulators could not be considered a suitable solution for those experimental settings, that require agents or distributed configurations.

Simulation based on virtual environments is a more effective approach especially when used in conjunction with agent-based and hardware virtualization technologies that allow to abstract physical resources and specific software components.

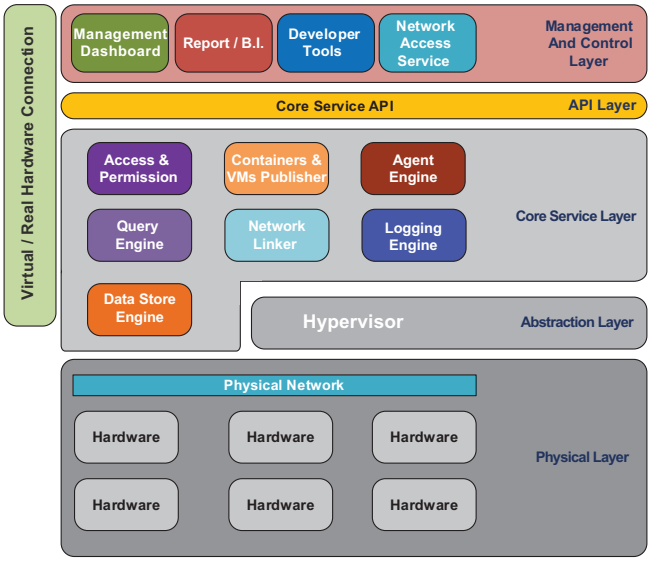


FIGURE 7.1: SMALLWORLD Architecture

The next subsection summarizes the main features of SMALLWORLD [82], a state-of-the-art virtual environment platform purposely designed for security assessment and education activities in the field of cybersecurity.

7.3.1 SMALLWORLD

SMALLWORLD is a software platform that has been devised in order to support the assessment, teaching and learning of security-related issues in various domains [82]. SMALLWORLD is based on state-of-the-art virtualization and cloud technologies for reproducing in a realistic setting a hybrid environment where large distributed computer systems can be deployed and from where they can interact with real life entities (users, software and hardware). SMALLWORLD enables security analysts and practitioners to design and enact complex scenarios which are dynamic and reactive and where a number of autonomous software agents can be deployed. SMALLWORLD agents are able to reproduce the behaviours of active entities of a given scenario, e.g. human users and/or malicious applications. This allows to the software components deployed into the virtual environment to behave and interact in a very realistic way with the actual real environment. For example, by suitably crafting the agents' behaviour, the scenario may evolve over time and produce unexpected and unpredictable events that are interesting to study and analyse through simulation logs.

Figure 7.1 depicts the SMALLWORLD architecture which is composed of five layers and has been designed to be extensible and hypervisor-independent.

The *Physical layer* hosts computational, storage and networking hardware configured in a suitable way in order to offer fault tolerance, business continuity and data replication mechanisms services for proper and scalable operation of the hypervisor.

The *Abstraction layer* virtualizes and hides hardware details which can then be easily changed/improved for scalability purposes without impacting on the overall system operations. This layer hosts the virtual machine monitor and the network hypervisor, which respectively enable to define via software the virtual computational nodes, along with the above operating systems and software layers (software defined systems) and the virtual network infrastructure (software-defined networking). There are many off-the-shelf hypervisor solutions that offers these features and

that can be employed in this layer. The current prototype of SMALLWORLD relies on Openstack [19, 177], however other implementations, i.e. VirtualBox [124], OpenNebula [153], are planned.

The *Core Service Layer* hosts the main software components that implement the core SMALLWORLD features which are then exposed by the overlying API layer. The blocks depicted inside this layer correspond to software components which realize specific SMALLWORLD functionalities. The *Network Linker* communicates with the underlying network hypervisor and introduces facilities to manage the networking services (i.e. routing, switching, bandwidth shaping, firewalling, policies). The *Publisher* is responsible to install applications (e.g. vulnerable software, malware, etc.) and agents in a given scenario. The *Datastore Engine* handles information that must be stored into suitable databases on the basis of the data type. This component does not use the abstraction layer. Data kept by the datastore engine are retrieved by the *Query Engine* and used by the *Management and Control Layer* to gather and compute statistics about the platform usage, e.g. users' and agents' activities, network bandwidth usage, traffic logs and other information.

The *Agent Engine* is basically an agent based [20, 210, 50] real-time simulation engine. It performs four main functions: (i) translates Agent Behaviour from a suitable agent description language (ADL) format to executable code; (ii) provides an API for deploying and planning all simulation steps; (iii) executes agents' behaviours in cooperation with each other providing an efficient messages delivery system; (iv) exposes an interface to extract efficiently simulation logs. A *Controller* entity permits to add *worker nodes* to the simulation each of which handles the execution of a little cluster of agents. The Controller orchestrates the behaviours of worker nodes.

The *API Layer* introduces a platform independent API which exposes the SMALLWORLD interface. This API is used for the implementation of the applications of the Management and Control Layer and it is fundamental for allowing the design and the development of reusable SMALLWORLD scenarios independently from the software technologies used in the underlying layers. The API is made available both as a Java framework and as a set of REST services.

The *Management and Control Layer* hosts a set of applications which ease the development and the management of SMALLWORLD scenarios and allow users and administrators to gather information about the status of the platform. In particular, the current version of SMALLWORLD provides the following tools. A *Dashboard*, enabling the management of scenarios, agents and virtual-machines. It also allows to display system usage and statics, set scenario parameters, handle user access and account management. A *Report* tool, which provides statistical data about the running scenarios. A set of *Development Tools* which include an agent development application and a scenario design tool.

SMALLWORLD can be exploited in various contexts and can be adapted to the specific available computational resources. Indeed, it provides different kinds of access and two type of installation: in site or in cloud. For example, an enterprise that has to deal with a large amount of data (e.g. VM images, system logs, etc.) into SMALLWORLD or does not want to expose private data and can afford a suitable hardware investment may opt for a in site deployment solution. The features to deliver to the client, and the respective cost, are fully customizable thanks to the modular design of the environment. On the other hand, SMALLWORLD can also be deployed on a cloud environment and made available as a service. This last solution allows the user to have immediate access to SMALLWORLD avoiding hardware investment and configuration efforts.

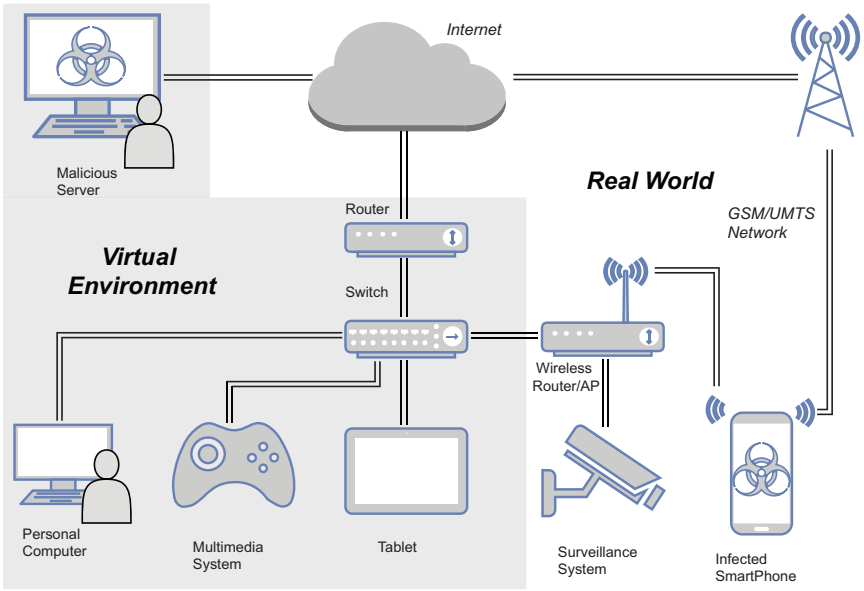


FIGURE 7.2: A typical insecure Smart Home scenario

7.4 Case study

This section illustrates three IoT scenarios that have been employed to investigate about the exploration and exploitation of common smart objects vulnerabilities. The scenarios were built by using the features of SMALLWORLD. The combined use of real devices interacting with a virtual environment allowed to analyse these IoT scenarios, assess their cybersecurity issues and conduct a suitable risk evaluation.

In particular, three variants of the same scenario were considered for studying the exposure to data leakage attacks and evaluate the effectiveness of two potential solutions.

The basic scenario is depicted in Figure 7.2 and it is typical of a smart home setting. The SMALLWORLD virtual environment hosts some nodes that interact with the real world outside through three distinct interfaces, two of which are connected to the Internet, and one to the local network of the smart home. A *malicious* node runs in a virtual machine and it is connected to the Internet through the first interface. Albeit it is located inside the virtual environment, it has no direct access to the other virtualized nodes which are connected to a trunk of the smart home LAN. The other nodes running in VMs are: a Personal Computer intended to be used for typical users's on-line activities (e.g. browsing web-pages) and to access the video surveillance system; a multimedia system and a Tablet. The tablet is emulated within the virtual environment through the use of a virtualized Android OS. These three nodes can access the Internet, through a switch which is in turn connected to a router acting as a gateway. They can also access the Smart Home LAN. For the purposes of the experiments, the connection to the switch is modelled as a cable connection (in a real setting it would have been a wireless connection), however this does not represent a limitation for the sake of the type of security assessment for which the scenario has been devised. Reproducing the behaviour of a wirelesses connection would have been useful only in the case of evaluation of attacks such as eavesdropping or sniffing on the physical channel.

The real devices involved in the scenario are: a smart surveillance camera and an *infected* Android smartphone. The camera is directly connected to the home

LAN. The smartphone can exploit two connections: one to the LAN through a WiFi access point and one to the Internet through a mobile network subscription (e.g. GSM/UMTS/LTE).

Finally, in order to animate the virtual portion of the scenario, two agents have been injected inside it: one in the VM running the Personal Computer and another in the malicious node. The former is in charge of browsing web pages and accessing the video surveillance system, the latter communicates with the infected devices sending them commands to accomplish and receiving the stolen information. The behaviours of these two agents were specified by means of state machines described in the SMALLWORLD agent description language. These state machines were designed and crafted on the basis of a domain expert's knowledge. In general the specification of agents behaviour come from a preliminary analysis of the entities they simulate. This holds in particular for malware applications that have to be first captured and then reverse engineered [39] or their behaviour inferred performing process mining activities on the logs of infected systems.

7.4.1 Attacking the video surveillance system

Smartphones are among the most common IoT devices and, because of their characteristics and features, represent a suitable attack vector. Such kind of devices interact everyday with different environments and establish connections not only with the home area network but also with dangerous access points as may be found in public networks. This situation, in some ways, mirrors the evolution of security on other platforms like the desktop PC, where, early attacks focused on the network layer and then migrated to the OS. An infected smartphone, which have access to a LAN of a smart home, can be easily used as the entry point to launch an attack, take control of other vulnerable IoT devices and perform malicious activities. There are many publicly available exploits for both iOS [56] and Android [9, 200] devices. It is possible to unlock both Apple iPhones, by means of so called *Jailbreaks* [115], and Android smartphones, by suitable *rooting* procedures [178], and then to install vulnerable software on them. In this work, Android has been chosen because it is easier to configure the OS components due to its open-source nature and to the size of the developer community.

In the proposed scenario the remote attacker exploits the Android Stagefright Integer Overflow vulnerability (described in [67]) in order to execute remote commands. To conduct a successful attack he adds to the payload of a multimedia message the binary code of a *backdoor* [15, 165]. The malicious code is sent to the victim by email, MMS, any other kind of instant messaging application or just as a link to a web page. Once the code is downloaded, it is executed as a background `telnet` service listening on port `1035` which performs tasks intended to steal information stored on the phone without the user being aware of it. The attacker remotely controls the malicious application by connecting to the backdoor port through which he can access a command shell prompt.

In the proposed scenario, when the infected smartphone connects to the home network, it starts a network scan in order to find the other IoT devices, gathering information like device model and firmware version. Such information are sent to a malicious Command and Control (c&c) server, which processes them in order to find exploitable vulnerabilities. The target of this experiment will be the Video Surveillance System, with the intent to retrieve sensitive information or to take control of it.

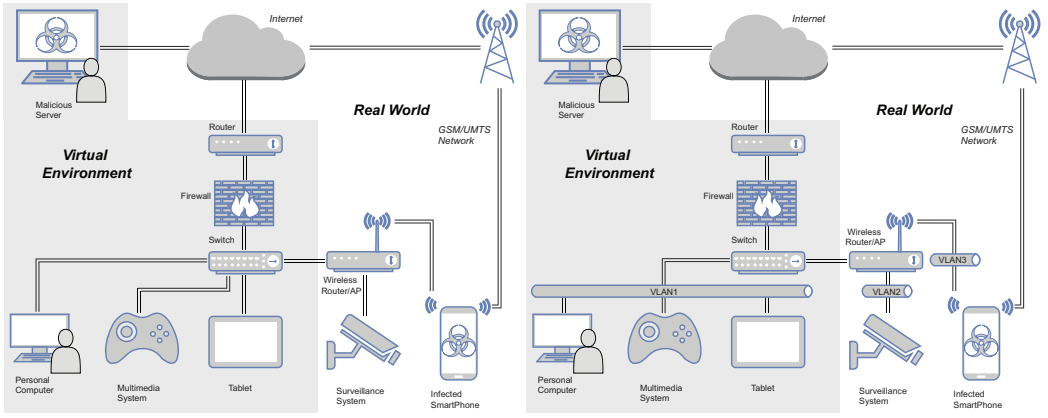


FIGURE 7.3: Scenario configuration with: (a) firewall, (b) separate VLANs

As the attack occurs in a LAN network, the attacker will instruct the compromised device to send spoofed Address Resolution Protocol (ARP) messages. The aim of this first phase of the attack is to associate the smartphone MAC address with the IP address of the default gateway, causing any traffic on the LAN to be sent to the attacker. Now, the attacker is able to inspect the packets and gather information, while forwarding the traffic to the actual default gateway to avoid discovery.

When the software agent placed on the personal computer try to access to the surveillance system through the web interface, it sends the credentials over the network without https encryption and the bad gateway can easily steal the credentials.

At this point the attacker, acting through the compromised device, can access the surveillance system and edit its configuration in order to make it accessible from the Internet. Considering that it is very common to find a telnet server running on such systems, often based on GNU/Linux distributions, the attacker can also login it with the stolen credentials and spawn a backdoor exploiting the ever-present `netcat` service [195].

Playing this scenario has shown how the attacker might gain sensitive information hurting the victim's privacy by means of which he can carry out criminal activities such as a blackmail. Moreover, having the control of the surveillance system, the attacker could study the victim's habits, understand when he is away from home and lead a successful robbery by turning off the home surveillance system.

7.4.2 Securing the smart home

The previous subsection has shown, thanks to the combined use of a virtual environment, software agents and real devices, how it is possible to reproduce a typical smart home scenario. This allowed to play inside it a real attack and to identify the security holes which are manly due to the malware ability to access the home LAN and, as a consequence, all the connected devices.

A first step towards securing the home network consists in the installation of a suitable configured firewall as depicted in Figure 7.3a. A simple provision consists in disallowing connections to the LAN which are initiated from the external. In this case the malware is confined inside the local network. It can perform the network scan as before but the Command and Control server cannot connect to the backdoor.

However, the smartphone does not only operates within the delimited zone. It can also access to the Internet through the mobile (GSM/UMTS/LTE) network. When the device is connected in such a way, the firewall is bypassed and the malicious application can send data to the malicious server in order to receive the commands that it will subsequently execute inside the LAN. This way, the attack continues as before with the difference that the attacker sends commands to and receives data from the smartphone when the device is connected to the mobile network. In addition, the malware application must enable a rule on the firewall in order to allow the attacker to exploit the backdoor. The experiments conducted showed that this countermeasure is not adequate to properly address this attack.

A third scenario has been considered for evaluating a suitable countermeasure to the described attack. The following steps can be considered as common best practices [126] to achieve a basilar home-protection:

- identify which devices need to be protected;
- group devices into logical groups;
- identify critical and not-critical groups;
- isolate each group in a separate sub-network and monitor activities that occur among them.

Each of the considered devices should be evaluated independently in order to find which one is part of critical assets. Being able to make such a separation, in many cases, is not an easy task. Indeed, the identification of critical devices is not only linked to critical assets to which they have direct access, but also to the logical dependencies that exist among them. Logical dependencies have a key role in this task, and if they are not suitably handled they can easily become *Trojan horses* through which access to critical assets can be gained. Based on such type of analysis, a logical separation among the devices, ensuring a suitable level of security, can be derived.

Following these guidelines it is easy to identify the surveillance system as the most critical device. As a consequence, it must necessarily be placed in a separate group and must have the least possible interaction with other devices. Another group is represented by *non mobile* devices, which in our scenario are confined exclusively within the home, i.e. the multimedia system, the tablet and the PC. Devices that can connect to other networks, such as the smartphone in the considered setting, must be part of a separate group because they are potential attack vectors. Another group, not present in this scenario, could be one reserved to host devices.

To isolate the logical groups identified in the previous phase, the home network is divided into three sub-networks, by resorting to the use of Virtual LANs, corresponding to the above-described devices groups. Using this configuration, the smartphone, which is the main attack vector, is isolated and cannot perform scanning in the whole network in order to discover the devices inside the smart home.

7.5 Summary

The pervasive diffusion of smart devices is going to change many aspects of our daily lives and also the way how most business activities will be accomplished in the next future. Just think how the smartphone already changed the way people communicate, make their appointments and plan their travels. The next step will

be to deal with the consequences of the worldwide spread of IoT devices. Because human beings are going to delegate many (critical) activities to smart devices, it is of utmost importance to suitably cope with cybersecurity issues threatening IoT. In this chapter we showed how simulation technologies can be effectively used for the assessment of cybersecurity scenarios involving IoT settings. In particular, the combined use of novel virtual environments, able to exploit state-of-the-art hardware virtualization technologies and cloud computing, agent-based simulation and real devices allow to design and evaluate, in a controlled way, IoT technologies (applications, protocols, device prototypes) and related security issues before releasing them in production. The effectiveness of the proposed approach was demonstrated through a case study regarding a typical smart home setting which was evaluated by means of the SMALLWORLD platform.

Chapter 8

Conclusions

The cyber security landscape is affected by a lot of significant problems, from developers that do not properly follow guidelines for the development of secure coding best practices to worldwide diffusion of malware toolkits that allow even users with little expertise to perform a cyber attack. A lot of efforts are required in order to lead the cyber security towards a better state, and that means adapting existing security solutions in response to the ever-changing security landscape and devising new solutions when needed. The research activities that we presented in this thesis find their place in this complex scenario.

Here we summarise the contributions, draw the conclusions of our work and discuss the future work. The chapter is concluded with the list of publications done during the Ph.D.

8.1 Summary of Conclusions

The research activity presented in this thesis aims at improving aspects of cyber security related to Hybrid Anomaly Detection Systems, Intrusion Detection Systems, Access Control Systems and Internet of Things, from a data analysis perspective and with particular emphasis on the concept of anomaly.

In Chapter 3 we presented *HALF*, a generic framework for anomaly detection. The framework has been designed to accommodate multiple mining algorithms of a specific domain by organizing them in a hierarchical structure in order to offer an higher and flexible detection capability. *HALF* can be easily employed in different application domains such as fraud detection, speech recognition and intrusion detection for cybersecurity, due to its generality. The framework fully supports the entire data analysis process, from the training and data collection phases to the execution of data analysis algorithms. Moreover, we introduced a learning module to handle the concept drift. We showed how *HALF* can be exploited in practice to implement a Network Intrusion Detection System and a Steganalysis tool, through two case studies. We explained the steps that need to be undertaken in order to verticalise the framework in the chosen application domain, from developing a module for parsing the data units to deploying and linking the chosen techniques. For the first case study we developed a system that can detect anomalies in network traffic of separate hosts and send e-mails to a network administrator to notify the detected attacks. In the second case study we configured the framework as a hybrid steganalysis tool able to find hidden content and report suspicious images.

In Chapter 4 we presented a novel anomaly-based intrusion detection technique, called PCkAD. PCkAD exploits the spatial distribution of the n-grams to model legitimate traffic. The technique learns the n-gram distribution in a semi-supervised fashion, in that it only requires legitimate samples for the training phase. Moreover, it uses four criteria to distinguish between different types of legitimate traffic. For

each type of traffic PCkAD builds a separate model to better capture the relationship between size and content inherent in the network traffic. A large experimental campaign was conducted to assess the performances and reliability against evasion attacks of PCkAD, whose main results are described in the following:

- The sensitivity analysis concerning the effect of the parameters' values on PCkAD confirmed that *i)* taking into account chunks is beneficial to the quality of the detection; *ii)* very short and long n-grams may lead to unsatisfactory performances, due to low discriminating power and higher risk of overfitting, respectively; *iii)* PCkAD is more robust to threshold variations when employing chunks with the values that guarantees the best performances.

- PCkAD was compared with other two IDS techniques based on n-grams on a variety of datasets. Our technique outperforms the other two in every case and obtains an AUC very close to 1. We believe that such results are due to the amount of structural information that PCkAD exploits to model network traffic.

- We evaluated the reliability of PCkAD against blending attacks and the experimental results show that the technique gets a significantly better accuracy when employing chunks, which means that it can guarantee a reliable detection.

We further investigated the usage of the n-gram technique in Chapter 5, where we discussed an approach for compressing n-gram-based classification models, in the context of intrusion detection. Specifically, our goal was to build compressed models that can classify content-based attacks in a satisfactory way. We explained the process that we employ the compressed model. The first step is to build a model containing all the n-grams learned from the training data and then compress it by resorting to the k-centre problem. We then compute the n-gram distribution within the legitimate relevant payloads, for each centre. During the classification phase, the anomaly score of a packet is computed by means of the concept of byte coverage. We evaluated our approach on a real word dataset to assess the performances of compressed models. We investigated the performances with respect to a few parameters, such as the compression level. The results are promising, we achieved satisfactory performances with significant configurations. In particular, a number of compressed models with a compression level less or equal than 10% obtained an AUC greater than 90% and even 95%.

In Chapter 6 we presented ML-AC, a machine learning empowered access control system designed to refine and update access policies in a dynamic fashion. The goal of ML-AC is to prevent insider threats and enable automated policy administration. The approach uses decision trees to generates new policy rules that capture runtime aspects concerning access control, in the form of user behaviours. These rules are exploited to detect legitimate users with malicious intentions. The rules are keep up-to-date by employing algorithms that can handle the concept drift. In addition, ML-AC resorts to association rule mining for detecting policy inconsistencies. We presented a case study to demonstrate how ML-AC works. The case study consists of four different scenarios that highlight the capabilities of our proposal. We then presented an experiment to show the impact of the classes of interaction when it comes to detect anomalous behaviours.

At last, in Chapter 7 we focused on a worrisome aspect of IoT, i.e. the worldwide spread of vulnerable smart devices. These devices are designed without properly taking security problems into account and, as a consequence, they have security

holes that allows malicious users to easily perpetrate attacks. This problem is caused by the presence of manufacturers that lack prior experience with networked devices and try to place their products into the market as soon as possible. In order to accomplish their goal they end up focusing only on the functionalities of the devices and overlooking security problems. In an effort to address the diffusion of vulnerable devices, researchers are focusing on the evaluation of security properties. Following this direction, we showed that by combining novel virtual environments, able to exploit state-of-the-art hardware virtualization technologies and cloud computing, agent-based simulation and real devices we can effectively assess cyber security scenarios involving IoT settings. We discussed a case study regarding a typical smart home setting to demonstrate the effectiveness of the proposed approach. Specifically, the SMALLWORLD platform was used for the evaluation. We considered three variants of the same scenario for studying the exposure to data leakage attacks and evaluate the effectiveness of two potential solutions. The first scenario presented represents a typical smart home setting consisting of both virtual and real devices. A malicious node is also present as a virtual device. The node is connected to the Internet through which it tries to compromise one or more devices that are part of the smart home network. In this scenario the remote attacker manages to compromise a vulnerable Android smartphone, by exploiting the Android Stagefright Integer Overflow vulnerability, and ultimately take control of the surveillance system. In the second scenario we evaluated a new network setting in which a firewall is installed and configured to prevent a remote user from initiating a connection from the external. However, the malicious user can still take advantage of the smartphone when it is connected to a mobile network. In the third and last scenario the smart home devices are placed in separate sub-networks, based on their logical dependencies and the level of criticality of the assets they can access to, in order to limit their interaction. As a result, the malicious user can no longer exploit the smartphone to access to the surveillance system.

8.2 Summary of contributions

In this section we recap the thesis contributions:

- We presented a flexible multi-domain framework, called *HALF*, that generalizes the problem of anomaly detection. The framework is designed to embrace both signature-based and anomaly-based techniques. It also allows the combination of different models for the analysis of data, by exploiting a hierarchical structure. *HALF* can process any kind of data and is not bound to any specific application domain, due to its generality. We described two case studies to show how the framework can be exploited in practice.

- We proposed a novel anomaly-based intrusion detection technique, called PCkAD. Unlike existing approaches, PCkAD builds models of legitimate traffic by learning the spatial distribution of the n-grams occurring in the relevant payloads of normal traffic. The strategy adopted makes the technique resilient to a well-known evasion attack, called blending, indeed we proved that performing such attack against PCkAD turns out to be NP-hard. PCkAD is also able to achieve a high detection rate with a very low false positive rate on a variety of datasets.

- We developed an approach to reduce the size of n-gram-based classification models, in the context of intrusion detection. The approach resorts to the k-center problem to select a representative subset of all the n-grams of a given model. A post training phase is then employed to capture the distribution of the n-grams in the training set. We then introduced the concept of byte coverage which is used to compute the anomaly score of a network packet. Experiments conducted on a real-world dataset show promising results.
- We designed a system, called ML-AC, which integrates ABAC with machine learning to autonomously refine and update access control policies. ML-AC exploits well-known machine learning techniques to automate generation and update of behavioural rules. We analysed a case study consisting of four different scenarios to show the capabilities of ML-AC.
- We proposed the combination of virtual environments, agent-based simulation and real devices as a mean to accurately evaluate and assess realistic IoT deployment scenarios which may involve complex networking infrastructures. We presented a case study describing a typical smart home setting to demonstrate the effectiveness of the proposed approach.

8.3 Open issues and future work

In what follows we discuss future work directions that arise from this thesis:

- *HALF* is steadily under development. We aim at providing new additional packages to support other domains and developing tools for monitoring and collection of statistics. Moreover, we want to finalise the development of the learning module that we introduced. The module is very important in that its goal is to handle the concept drift and makes techniques that work with static models suitable for contexts where the nature of data change over time. We also intend to conduct an extensive campaign of experiments to validate the effectiveness of the framework.
- Concerning the compression of n-gram-based models, a number of interesting research directions can be explored. We intend to investigate an ensemble approach based on these models in order to get even better performances. Indeed, such models can be seen as weak classifiers when compared to the original ones. We would like to explore new strategies to compute both the n-gram distribution for each centre and the anomaly score. We also aim at evaluating the compressed models on other datasets and exploring the impact of the compressed models on the processing speed of the approach.
- Another research direction of interest is the evaluation of the model compression in PCkAD. In this case, the problem seems to be harder because PCkAD relies on the spatial distribution of the n-grams. The payloads are split in chunks, so it is not clear how the set of centres should be built. We want to make sure that all the chunks are properly represented by the centres. If there were chunks whose n-grams are not represented there would be a considerable information loss. It would be also interesting to investigate how to take advantage of the n-gram spatial distribution to discover patterns that characterise malicious observations and turn them into signatures that could be stored in the knowledge base of a signature-based IDS.

- Regarding ML-AC, there are a number of interesting aspects to address. ML-AC still lacks a toolchain to automate the policy rule generation. This addition is our highest priority as a future work. We also intend to perform more extensive esperiments with a real-life dataset to evaluate the effectiveness of ML-AC. Another aspect concerns the classes of interaction. It might not always be possible to clearly define the classes based only on the knowledge derived from the context and the access control policies. There may be some classes that should be merged or split, hence we are considering to design a preprocessing step whose goal is to support the classes definition process. ML-AC employs different machine learning algorithms to achieve its goals, therefore it may incur in the risk of being subject to adversarial attacks. We would like to study the adversarial attacker models that might hinder the normal operational state of ML-AC to devise practical mitigation strategies against them.

8.4 List of publications

This section describes the publications and contributions to journals and conferences that has resulted from the research work done during the Ph.D.

8.4.1 Journals

- Furfaro, A., Argento, L., Parise, A., & Piccolo, A. (2017). Using virtual environments for the assessment of cybersecurity issues in IoT scenarios. *Simulation Modelling Practice and Theory, 73,* 43-54.

- Angiulli, F., Argento, L., & Angelo, F. (2017). Exploiting content spatial distribution to improve detection of intrusions. *Accepted on September 15th 2017, in ACM Transactions on Internet Technology*.

- Furfaro, A., Parise, A., Argento, L., Piccolo, A., & Sacca', D (2017). A Cloud-based Platform for the Emulation of Complex Cybersecurity Scenarios. *Under review*.

- Angiulli, F., Argento, L., Furfaro, A., & Parise, A. (2018). A Hierarchical Hybrid Framework for Modelling Anomalous Behaviours. *Simulation Modelling Practice and Theory, 82,* 103-115.

8.4.2 Conferences

- Argento, L., & Furfaro, A. (2015, May). A multi-protocol framework for the development of collaborative virtual environments. In *Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on* (pp. 449-454). IEEE.

- Angiulli, F., Argento, L., & Furfaro, A. (2015, November). Exploiting n-gram location for intrusion detection. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on* (pp. 1093-1098). IEEE.

Bibliography

[1] Elizabeth Shawt Adams. "A Study of Trigrams and Their Feasibility As Index Terms in a Full Text Information Retrieval System". UMI Order No. GAX92-12700. PhD thesis. Washington, DC, USA, 1992.

[2] Sebastian Albrecht et al. "Generalized radial basis function networks for classification and novelty detection: self-organization of optimal Bayesian decision". In: *Neural Networks* 13.10 (2000), pp. 1075–1093.

[3] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. "Just in time classifiers: Managing the slow drift case". In: *Proceedings of the 2009 International Joint Conference on Neural Networks*. IJCNN'09. Atlanta, GA, USA: IEEE, 2009, pp. 1537–1543.

[4] Mahdi Alizadeh et al. "Behavior Analysis in the Medical Sector: Theory and Practice". In: *Proceedings of Symposium on Applied Computing*. ACM, 2018.

[5] Palo Alto. *Next-Generation Firewall*. https://www.paloaltonetworks.com/products/secure-the-network/next-generation-firewall. 2017.

[6] Waqas Aman and Einar Snekkenes. "Managing security trade-offs in the Internet of Things using adaptive security". In: *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE. 2015, pp. 362–368. DOI: 10.1109/icitst.2015.7412122.

[7] Nadav Amit et al. "Bare-metal performance for virtual machines with exitless interrupts". In: *Communications of the ACM* 59.1 (2015), pp. 108–116. DOI: 10.1145/2845648.

[8] Brandie Anderson et al. *Cyber Risk Report 2016*. Tech. rep. Hewlett Packard Enterprise, 2016.

[9] *AndroidVurnelrabilities.org*. http://androidvulnerabilities.org/all.

[10] Artur Andrzejak, Felix Langner, and Silvestre Zabala. "Interpretable models from distributed data via merging of decision trees". In: *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*. IEEE. 2013, pp. 1–9.

[11] Fabrizio Angiulli, Luciano Argento, and Angelo Furfaro. "Exploiting N-Gram Location for Intrusion Detection". In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. Vietri sul Mare, Salerno, Italia: IEEE, 2015, pp. 1093–1098. DOI: 10.1109/ictai.2015.155.

[12] Fabrizio Angiulli, Luciano Argento, and Angelo Furfaro. *PCkAD source code*. https://github.com/F3nDis/PCkAD. 2017.

[13] Sabeel Ansari, SG Rajeev, and HS Chandrashekar.
"Packet sniffing: a brief introduction".
In: *IEEE potentials* 21.5 (2002), pp. 17–19.

[14] Manos Antonakakis et al. "Understanding the mirai botnet".
In: *USENIX Security Symposium*. 2017.

[15] Iván Arce. "The Shellcode Generation".
In: *IEEE Security & Privacy Magazine* 2.5 (Sept. 2004), pp. 72–76.
ISSN: 1540-7993. DOI: 10.1109/msp.2004.87.

[16] Mehran Asadi, Christopher Zimmerman, and Afrand Agah.
"A Game-theoretic Approach to Security and Power Conservation in Wireless Sensor Networks".
In: *International Journal of Network Security* 15.1 (2013), pp. 50–58.

[17] Ahmad W. Atamli and Andrew Martin.
"Threat-Based Security Analysis for the Internet of Things".
In: *2014 International Workshop on Secure Internet of Things*. IEEE. 2014, pp. 35–43. DOI: 10.1109/siot.2014.10.

[18] Luigi Atzori, Antonio Iera, and Giacomo Morabito.
"The Internet of things: A survey".
In: *Computer Networks* 54.15 (2010), pp. 2787–2805.
DOI: 10.1016/j.comnet.2010.05.010.

[19] Guillaume Aubuchon et al. *Expediting Digital Workflow with OpenStack*.
White Paper. 2015.
URL: https://www.openstack.org/assets/pdf-downloads/OpenStack-Workflow-White-Paper-Letter-Final.pdf.

[20] Robert Axelrod. *The complexity of cooperation: Agent-based models of competition and collaboration*. Walter de Gruyter GmbH, 1997.
DOI: 10.1515/9781400822300.

[21] Stefan Axelsson. *Intrusion detection systems: A survey and taxonomy*. Tech. rep. 2000.

[22] M Ali Aydın, A Halim Zaim, and K Gökhan Ceylan. "A hybrid intrusion detection system design for computer network security".
In: *Computers & Electrical Engineering* 35.3 (2009), pp. 517–526.

[23] Sachin Babar et al. "Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT)".
In: *Recent Trends in Network Security and Applications*. Springer, 2010, pp. 420–429. DOI: 10.1007/978-3-642-14478-3_42.

[24] Daniel Barbará et al. "ADAM: Detecting Intrusions by Data Mining".
In: *In Proceedings of the IEEE Workshop on Information Assurance and Security*. IEEE, 2001, pp. 11–16.

[25] Lujo Bauer, Scott Garriss, and Michael K Reiter. "Detecting and resolving policy misconfigurations in access-control systems". In: *ACM Transactions on Information and System Security (TISSEC)* 14.1 (2011), p. 2.

[26] Chakib Bekara.
"Security Issues and Challenges for the IoT-based Smart Grid".
In: *Procedia Computer Science* 34 (2014), pp. 532–537.
DOI: 10.1016/j.procs.2014.07.064.

[27] Salem Benferhat, Tayeb Kenaza, and Aicha Mokhta2ri.
"A naive bayes approach for detecting coordinated attacks".
In: *IEEE Int. Conf. on Computer Software and Applications (COMPSAC)*. 2008, pp. 704–709.

[28] Lon A. Berk and Sergio F. Oehninger.
The Risk of Insuring Supply Chains From Cyber Risk. Law360. 2015.
URL: http://www.law360.com/articles/668004/the-risk-of-insuring-supply-chains-from-cyber-risk.

[29] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita.
"Network anomaly detection: methods, systems and tools".
In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), pp. 303–336.

[30] Battista Biggio, Giorgio Fumera, and Fabio Roli.
"Security evaluation of pattern classifiers under attack". In: *IEEE transactions on knowledge and data engineering* 26.4 (2014), pp. 984–996.

[31] Battista Biggio et al. "One-and-a-Half-Class Multiple Classifier Systems for Secure Learning Against Evasion Attacks at Test Time".
In: *International Workshop on Multiple Classifier Systems*. Springer. 2015, pp. 168–180.

[32] Battista Biggio et al. "Security evaluation of biometric authentication systems under real spoofing attacks". In: *IET biometrics* 1.1 (2012), pp. 11–24.

[33] Leyla Bilge et al. "Disclosure: detecting botnet command and control servers through large-scale netflow analysis".
In: *ACM Inf. Conf. on Computer Security Applications*. 2012, pp. 129–138.

[34] Leyla Bilge et al. "EXPOSURE: a passive DNS analysis service to detect and report malicious domains". In: *ACM Transactions on Information and System Security (TISSEC)* 16.4 (2014), p. 14.

[35] Leyla Bilge et al.
"EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis."
In: *Ndss*. 2011.

[36] Misty Blowers and Jonathan Williams.
"Machine learning applied to cyber operations".
In: *Network Science and Cybersecurity*. Springer, 2014, pp. 155–175.

[37] Nathaniel Boggs et al. "Synthetic Data Generation and Defense in Depth Measurement of Web Applications".
In: *Research in Attacks, Intrusions and Defenses*. Springer, 2014, pp. 234–254.

[38] Damiano Bolzoni, Sandro Etalle, and Pieter Hartel.
"POSEIDON: a 2-tier anomaly-based network intrusion detection system".
In: *Proc. of IEEE Int. Workshop on Information Assurance (IWIA)*. 2006, pp. 144–156.

[39] Michele Bombardieri et al.
"Honeypot-Powered Malware Reverse Engineering".
In: *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. Institute of Electrical and Electronics Engineers (IEEE), 2016.
DOI: 10.1109/ic2ew.2016.16.

[40] L Breiman. "Bagging predictors. Univ. California Technical Report No. 421".
In: (1994).

[41] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[42] Peter F Brown et al. “Class-based n-gram models of natural language”. In: *Computational linguistics* 18.4 (1992), pp. 467–479.

[43] Michael Brückner, Christian Kanzow, and Tobias Scheffer. “Static prediction games for adversarial learning problems”. In: *Journal of Machine Learning Research* 13.Sep (2012), pp. 2617–2654.

[44] Anna L Buczak and Erhan Guven. “A survey of data mining and machine learning methods for cyber security intrusion detection”. In: *IEEE Communications Surveys & Tutorials* 18.2 (2016), pp. 1153–1176.

[45] Gail A Carpenter et al. “Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps”. In: *Neural Networks, IEEE Transactions on* 3.5 (1992), pp. 698–713.

[46] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[47] Kavitha Chandrasekar et al. *Internet Security Threat Report ISTR*. Tech. rep.

[48] Dustin Childs et al. *Cyber Risk Report 2015*. Tech. rep. Hewlett Packard Enterprise, 2015.

[49] Franco Cicirelli, Angelo Furfaro, and Libero Nigro. “Exploiting agents for modelling and simulation of coverage control protocols in large sensor networks”. In: *Journal of Systems and Software* 80.11 (2007), pp. 1817–1832. DOI: 10.1016/j.jss.2007.02.015.

[50] Franco Cicirelli, Angelo Furfaro, and Libero Nigro. “Modelling and simulation of complex manufacturing systems using statechart-based actors”. In: *Simulation Modelling Practice and Theory* 19.2 (2011), pp. 685–703. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2010.10.010.

[51] Franco Cicirelli et al. “HLA_ACTOR_REPAST: An approach to distributing RePast models for high-performance simulations”. In: *Simulation Modelling Practice and Theory* 19.1 (2011), pp. 283–300. DOI: 10.1016/j.simpat.2010.06.013.

[52] Cisco. *Snort*. https://www.snort.org/.

[53] Christopher Budd (Global Threat Communications).

[54] Elisa Costante et al. “A Hybrid Framework for Data Loss Prevention and Detection”. In: *Proceedings of IEEE Security and Privacy Workshops*. IEEE, 2016, pp. 324–333.

[55] Gabriela F Cretu et al. “Casting out demons: Sanitizing training data for anomaly sensors”. In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE. 2008, pp. 81–95.

[56] CVE Details. *Apple, Iphone Os: Security Vulnerabilities*. URL: http://www.cvedetails.com/vulnerability-list/vendor_id-49/product_id-15556/Apple-Iphone-Os.html.

[57] Nilesh Dalvi et al. “Adversarial classification”. In: *Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*. ACM. 2004, pp. 99–108.

[58] D. Dasgupta and N. Attoh-Okine. "Immunity-based systems: A survey".
In: *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*. Vol. 1.
Orlando, FL, USA: IEEE, 1997, pp. 369–374.
DOI: 10.1109/icsmc.1997.625778.

[59] Dipankar Dasgupta.
"Immunity-based intrusion detection system: a general framework".
In: *Proc. of the 22nd National Information Systems Security Conference*. Vol. 1.
Arlington, Virginia, USA, 1999, pp. 147–160.

[60] Jonathan J Davis and Andrew J Clark. "Data preprocessing for anomaly based network intrusion detection: A review".
In: *Computers & Security* 30.6 (2011), pp. 353–375.

[61] John D Day and Hubert Zimmermann. "The OSI reference model".
In: *Proceedings of the IEEE* 71.12 (1983), pp. 1334–1340.

[62] Francesco De Comité et al.
"Positive and unlabeled examples help learning".
In: *Algorithmic Learning Theory*. Springer. 1999, pp. 219–230.

[63] Ozgur Depren et al. "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks".
In: *Expert systems with Applications* 29.4 (2005), pp. 713–722.

[64] Theo Detristan et al. *Polymorphic shellcode engine using spectrum analysis*. 2003.

[65] Culture Media Department for Digital and Sport.
Secure by Design: Improving the cyber security of consumer Internet of Things.
Report. 2018. URL: https://iotsecurity2018.ca/wp-content/uploads/2018/04/Secure_by_Design_Report_.pdf.

[66] Pedro Domingos and Geoff Hulten. "Mining high-speed data streams".
In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*.
Boston, MA, USA: Association for Computing Machinery (ACM), 2000, pp. 71–80. DOI: 10.1145/347090.347107.

[67] Joshua Drake. "Stagefright: Scary Code in the Heart of Android".
In: *Black Hat USA*. Mandalay Bay, Las Vegas, NV, USA, Aug. 2015.
URL: https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-Android.pdf.

[68] Sumeet Dua and Xian Du. *Data mining and machine learning in cybersecurity*.
Boca Raton, FL: CRC press, 2016.

[69] Haimonti Dutta et al. "Distributed Top-K Outlier Detection from Astronomy Catalogs using the DEMAC System".
In: *Proceedings of the 2007 SIAM International Conference on Data Mining*.
Minneapolis,MN, USA: SIAM, 2007, pp. 473–478.
DOI: 10.1137/1.9781611972771.47.

[70] FY Edgeworth. "Xli. on discordant observations".
In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 23.143 (1887), pp. 364–375.

[71] Ivo Van der Elzen and Jeroen van Heugten.
"Techniques for detecting compromised IoT devices".
In: *University of Amsterdam* (2017).

[72] F-Secure. *Vienna Threat description*.
https://www.f-secure.com/v-descs/vienna.shtml. 2017.

[73] Wei Fan et al. "Using artificial anomalies to detect unknown and known network intrusions".
In: *Knowledge and Information Systems* 6.5 (2004), pp. 507–527.

[74] Hany Farid.
"Detecting hidden messages using higher-order statistical models".
In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 2. IEEE. Rochester, NY: IEEE, 2002, pp. 905 –908.
DOI: 10.1109/ICIP.2002.1040098.

[75] Tom Fawcett and Foster Provost. "Adaptive fraud detection".
In: *Data mining and knowledge discovery* 1.3 (1997), pp. 291–316.

[76] Adrienne Porter Felt et al. "Measuring HTTPS Adoption on the Web".
In: (2017).

[77] Prahlad Fogla et al. "Polymorphic Blending Attacks".
In: *Proc. of 15th USENIX Security Symposium*. Vancouver, B.C., Canada, 2006, pp. 241–256.

[78] Behrouz A Forouzan. *TCP/IP protocol suite*. McGraw-Hill, Inc., 2002.

[79] Vanessa Frias-Martinez, Salvatore J Stolfo, and Angelos D Keromytis.
"Behavior-based network access control: A proof-of-concept".
In: *International Conference on Information Security*. Springer. 2008, pp. 175–190.

[80] Vanessa Frias-Martinez et al.
"A network access control mechanism based on behavior profiles".
In: *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE. 2009, pp. 3–12.

[81] Richard M. Fujimoto. *Parallel and Distributed Simulation Systems*.
John Wiley & Sons, Inc., 2000.

[82] Angelo Furfaro et al. "A Virtual Environment for the Enactment of Realistic Cyber Security Scenarios". In: *Proc. of 2nd IEEE International Conference on Cloud Computing Technologies and Applications (CloudTech 2016)*.
Marrakesh, Morocco, 2016.

[83] John Gallant, David Maier, and James Astorer.
"On finding minimal length superstrings".
In: *Journal of Computer and System Sciences* 20.1 (1980), pp. 50 –58.

[84] Erich Gamma et al.
Design Patterns: Elements of Reusable Object-oriented Software.
Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.
ISBN: 0-201-63361-2.

[85] Pedro Garcia-Teodoro et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges".
In: *Computers & Security* 28.1 (2009), pp. 18–28.

[86] Gartner. *Gartner Says 6.4 Billion Connected "Thing" Will Be in Use in 2016, Up 30 Percent From 2015*.
http://www.gartner.com/newsroom/id/3165317. 2015.

[87] Mengmeng Ge and Dong Seong Kim. "A Framework for Modeling and Assessing Security of the Internet of Things". In: *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*.
Institute of Electrical & Electronics Engineers (IEEE), 2015, pp. 776–781.
DOI: 10.1109/icpads.2015.102.

[88] Béla Genge and Christos Siaterlis. "Developing cyber-physical experimental capabilities for the security analysis of the future Smart Grid".
In: *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*.
Manchester: Institute of Electrical & Electronics Engineers (IEEE), 2011, pp. 1–7. DOI: 10.1109/isgteurope.2011.6162766.

[89] Hamza Ghani et al. "Assessing the Security of Internet Connected Critical Infrastructures (The CoMiFin Project Approach)".
In: *Proc. of the Workshop on Security of the Internet of Things*. 2010.

[90] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli.
Fundamentals of Software Engineering. 2nd. Pearson, 2002.

[91] Charles H Giancarlo. *Packet switching network*. US Patent 4,768,190. 1988.

[92] Joshua Glasser and Brian Lindauer.
"Bridging the gap: A pragmatic approach to generating insider threat data".
In: *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE. 2013, pp. 98–104.

[93] Amir Globerson and Sam Roweis.
"Nightmare at test time: robust learning by feature deletion".
In: *Proceedings of the 23rd international conference on Machine learning*.
ACM. 2006, pp. 353–360.

[94] Teofilo F Gonzalez.
"Clustering to minimize the maximum intercluster distance".
In: *Theoretical Computer Science* 38 (1985), pp. 293–306.

[95] Alexander Gruenstein, Chao Wang, and Stephanie Seneff.
"Context-sensitive statistical language modeling".
In: *Ninth European Conference on Speech Communication and Technology*. 2005.

[96] Daniel Halperin et al. "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses".
In: *2008 IEEE Symposium on Security and Privacy (SP 2008)*. IEEE. 2008, pp. 129–142. DOI: 10.1109/sp.2008.31.

[97] Steve Hand. "High-performance virtualization: are we done?"
In: *Communications of the ACM* 59.1 (2015), pp. 107–107.
DOI: 10.1145/2845910.

[98] Jeremiah J. Harmsen and William A. Pearlman.
"Steganalysis of additive-noise modelable information hiding".
In: *Security and Watermarking of Multimedia Contents V*.
Ed. by Edward J. Delp III and Ping W. Wong.
Santa Clara, CA: SPIE-Intl Soc Optical Eng, June 2003, pp. 131–142.
DOI: 10.1117/12.476813.

[99] John A Hartigan and JA Hartigan. *Clustering algorithms*. Vol. 209. Wiley New York, 1975.

[100] Tin Kam Ho. "Random decision forests". In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 1. IEEE. 1995, pp. 278–282.

[101] T Ryan Hoens, Robi Polikar, and Nitesh V Chawla. "Learning from streaming data with concept drift and imbalance: an overview". In: *Progress in Artificial Intelligence* 1.1 (2012), pp. 89–101.

[102] Ning Hu, Phillip G Bradford, and Jun Liu. "Applying role based access control and genetic algorithms to insider threat detection". In: *Proceedings of the 44th annual Southeast regional conference*. ACM. 2006, pp. 790–791.

[103] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo. "Attribute-Based Access Control". In: *IEEE Computer* 48.2 (2015), pp. 85–88.

[104] Ling Huang et al. "Adversarial machine learning". In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. ACM. 2011, pp. 43–58.

[105] Xiaoqiu Huang et al. "PCAP: a whole-genome assembly program". In: *Genome research* 13.9 (2003), pp. 2164–2170.

[106] Zhexue Huang. "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values". In: *Data Mining and Knowledge Discovery* 2.3 (1998), pp. 283–304. ISSN: 1573-756X. DOI: 10.1023/A:1009769707641. URL: https://doi.org/10.1023/A:1009769707641.

[107] Matthias Hummer et al. "Adaptive identity and access management contextual data based policies". In: *EURASIP Journal on Information Security* 2016.1 (2016), p. 19.

[108] JeeHyun Hwang et al. "Mining likely properties of access control policies via association rule mining". In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer. 2010, pp. 193–208.

[109] IETF. *HTTP Over TLS*. https://tools.ietf.org/html/rfc2818. 2000.

[110] IETF. *Hypertext Transfer Protocol – HTTP/1.1*. https://tools.ietf.org/html/rfc2616. 1999.

[111] Imperva. *Web application attack report 5*. Tech. rep.

[112] Kenneth L. Ingham and Hajime Inoue. "Comparing anomaly detection techniques for http". In: *Int. Symp. on Recent Advances in Intrusion Detection (RAID)*. 2007, pp. 42–62.

[113] Rhena Inocencio. *BASHLITE Affects Devices Running on BusyBox*. https://blog.trendmicro.com/trendlabs-security-intelligence/bashlite-affects-devices-running-on-busybox/. 2014.

[114] SANS Institute. *Privacy and the Internet of Things*. White paper. 2017. URL: https://www.sans.org/reading-room/whitepapers/privacy/privacy-internet-things-38105.

[115] *iOS Jailbreak*. http://www.ios9cydia.com/.

[116] Fred Jelinek. "Self-organized language modeling for speech recognition". In: *Readings in speech recognition* (1990), pp. 450–506.

[117] Ming Jiang et al. "Steganalysis of Halftone Images". In: *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 2. Philadelphia, PA: IEEE, 2005, pp. 793–796. DOI: 10.1109/icassp.2005.1415524.

[118] Neil F Johnson and Sushil Jajodia. "Steganalysis of images created using current steganography software". In: *International Workshop on Information Hiding*. Vol. 1525. LNCS. Portland, Oregon: Springer, 1998, pp. 273–289.

[119] John Felix Charles Joseph et al. "CARRADS: Cross layer based adaptive real-time routing attack detection system for MANETS". In: *Computer Networks* 54.7 (2010), pp. 1126–1141.

[120] Antti Juvonen and Tuomo Sipola. "Adaptive framework for network traffic classification using dimensionality reduction and clustering". In: *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on*. IEEE. 2012, pp. 274–279.

[121] Stefan Katzenbeisser and Fabien Petitcolas. *Information hiding techniques for steganography and digital watermarking*. Norwood, MA, USA: Artech house, 2000.

[122] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. "A new intrusion detection system using support vector machines and hierarchical clustering". In: *The VLDB Journal* 16.4 (2007), pp. 507–521.

[123] Hahnsang Kim, Joshua Smith, and Kang G Shin. "Detecting energy-greedy anomalies and mobile malware variants". In: *Proceeding of the 6th international conference on Mobile systems, applications, and services - MobiSys'08*. Breckenridge, CO, USA: ACM, 2008, pp. 239–252. DOI: 10.1145/1378600.1378627.

[124] Gregory King. *Oracle VM 3: Building a Demo Environment using Oracle VM VirtualBox*. http://www.oracle.com/technetwork/server-storage/vm/ovm3-demo-vbox-1680215.pdf. 2012.

[125] Amit Klein. *Exploiting the XmlHttpRequest object in IE*. http://www.securityfocus.com/archive/1/411585. 2005.

[126] Eric D. Knapp and Joel Thomas Langill. *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress Publishing, 2011. ISBN: 9781597496469.

[127] Reinhard Kneser and Hermann Ney. "Improved clustering techniques for class-based statistical language modelling." In: *Eurospeech*. Vol. 93. 1993, pp. 973–76.

[128] Levent Koc, Thomas A Mazzuchi, and Shahram Sarkani. "A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier". In: *Expert Systems with Applications* 39.18 (2012), pp. 13492–13500.

[129] WC Koeppen, E Pilger, and R Wright. "Time series analysis of infrared satellite data for detecting thermal anomalies: a hybrid approach". In: *Bulletin of Volcanology* 73.5 (2011), pp. 577–593.

[130] Jeremy Z Kolter and Marcus A Maloof. "Learning to detect malicious executables in the wild". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 470–478.

[131] Christian Kreibich and Jon Crowcroft. "Honeycomb: creating intrusion detection signatures using honeypots". In: *ACM SIGCOMM computer communication review* 34.1 (2004), pp. 51–56.

[132] Diego Kreutz et al. "Software-Defined Networking: A Comprehensive Survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76. DOI: 10.1109/jproc.2014.2371999.

[133] Sachin Kumar et al. "A hybrid prognostics methodology for electronic products". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Hong Kong: IEEE, 2008, pp. 3479–3485. DOI: 10.1109/ijcnn.2008.4634294.

[134] Sathish Alampalayam Kumar, Tyler Vealey, and Harshit Srivastava. "Security in Internet of Things: Challenges, Solutions and Future Directions". In: *49th Hawaii International Conference on System Sciences (HICSS)*. Institute of Electrical & Electronics Engineers (IEEE), 2016, pp. 5772–5781. DOI: 10.1109/hicss.2016.714.

[135] Rikard Laxhammar. "Anomaly detection for sea surveillance". In: *Information Fusion, 2008 11th International Conference on*. Cologne, Germany: IEEE, 2008, pp. 1–8.

[136] Aleksandar Lazarevic et al. "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. San Francisco, CA, USA: SIAM, 2003, pp. 25–36. DOI: 10.1137/1.9781611972733.3.

[137] Dr. Jane LeClair. *Small Business, Big Threat: Protecting Small Businesses from Cyber Attacks*. Tech. rep. National Cybersecurity Institute.

[138] Philip A Legg et al. "Caught in the act of an insider attack: Detection and assessment of insider threat". In: *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*. IEEE. 2015, pp. 1–6.

[139] Wolfgang Leister et al. "An evaluation framework for adaptive security for the IoT in eHealth". In: *International Journal on Advances in Security* 7.3-4 (2014), pp. 93–109.

[140] Fabien Letouzey, François Denis, and Rémi Gilleron. "Learning from positive and unlabeled examples". In: *Algorithmic Learning Theory*. Springer. 2000, pp. 71–85.

[141] Yinhui Li et al. "An efficient intrusion detection system based on support vector machines and gradually feature removal method". In: *Expert Systems with Applications* 39.1 (2012), pp. 424–430.

[142] Richard Lippmann et al.
"The 1999 DARPA off-line intrusion detection evaluation".
In: *Computer networks* 34.4 (2000), pp. 579–595.

[143] Jing Liu, Yang Xiao, and C.L. Philip Chen.
"Authentication and Access Control in the Internet of Things". In: *2012 32nd International Conference on Distributed Computing Systems Workshops*.
Institute of Electrical & Electronics Engineers (IEEE), 2012, pp. 588–592.
DOI: 10.1109/icdcsw.2012.23.

[144] Siwei Lyu and Hany Farid.
"Steganalysis using higher-order image statistics". In: *IEEE transactions on Information Forensics and Security* 1.1 (2006), pp. 111–119.

[145] Matthew V Mahoney.
"Network traffic anomaly detection based on packet bytes".
In: *Proceedings of the 2003 ACM symposium on Applied computing*. ACM. 2003, pp. 346–350.

[146] Matthew V Mahoney and Philip K Chan.
PHAD: Packet header anomaly detection for identifying hostile network traffic.
Tech. rep. CS-2001-04. Florida Institute of Technology, 2001. URL: https://cs.fit.edu/media/TechnicalReports/cs-2001-04.pdf.

[147] Marcus Maloof and Gregory Stephens.
"Elicit: A system for detecting insiders who violate need-to-know".
In: *Recent Advances in Intrusion Detection*. Springer. 2007, pp. 146–166.

[148] Friedemann Mattern and Christian Floerkemeier.
"From the Internet of Computers to the Internet of Things".
In: *From active data management to event-based systems and more*. Vol. 6462.
LNCS. Springer, 2010, pp. 242–259.
DOI: 10.1007/978-3-642-17226-7_15.

[149] Orestis Mavropoulos et al.
"ASTo: A tool for security analysis of IoT systems".
In: *Software Engineering Research, Management and Applications (SERA), 2017 IEEE 15th International Conference on*. IEEE. 2017, pp. 395–400.

[150] Brent T McBride, Gilbert L Peterson, and Steven C Gustafson.
"A new blind method for detecting novel steganography".
In: *Digital Investigation* 2.1 (2005), pp. 50–70.

[151] D Michie et al. *Quinlan, JR C4. 5: Programs for Machine Learning*. 1993.

[152] Charlie Miller and Chris Valasek.
"Remote exploitation of an unaltered passenger vehicle". In: *Black Hat USA*.
Mandalay Bay, Las Vegas, NV, USA, 2015.

[153] Dejan Milojičić, Ignacio M. Llorente, and Ruben S. Montero.
"OpenNebula: A Cloud Management Tool".
In: *IEEE Internet Computing* 15.2 (2011), pp. 11–14. ISSN: 1089-7801.
DOI: 10.1109/mic.2011.44.

[154] Leandro L Minku and Xin Yao.
"DDD: A new ensemble approach for dealing with concept drift". In: *IEEE transactions on knowledge and data engineering* 24.4 (2012), pp. 619–633.

[155] Jelena Mirkovic et al. “Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)”. In: (2004).

[156] MITRE Corporation. *Common Vulnerabilities and Exposures*. CVE 2012-0911. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0911, 2012.

[157] MITRE Corporation. *Common Vulnerabilities and Exposures*. CVE 2014-6271. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271, 2014.

[158] David Moore et al. “Inferring internet denial-of-service activity”. In: *ACM Transactions on Computer Systems (TOCS)* 24.2 (2006), pp. 115–139.

[159] Ricardo Neisse et al. “SecKit: A Model-based Security Toolkit for the Internet of Things”. In: *Computers & Security* 54 (2015), pp. 60–76. DOI: 10.1016/j.cose.2015.06.002.

[160] Suraj Nellikar, David M Nicol, and Jai J Choi. “Role-based differentiation for insider detection algorithms”. In: *Proceedings of the 2010 ACM workshop on Insider threats*. ACM. 2010, pp. 55–62.

[161] Blaine Nelson et al. “Misleading learners: Co-opting your spam filter”. In: *Machine learning in cyber trust*. Springer, 2009, pp. 17–51.

[162] Arooj Nissar and AH Mir. “Classification of steganalysis techniques: A study”. In: *Digital Signal Processing* 20.6 (2010), pp. 1758–1770.

[163] OASIS XACML TC. *eXtensible Access Control Markup Language (XACML) version 3.0*. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. 2013.

[164] Hilarie Orman. “The Morris worm: A fifteen-year perspective”. In: *IEEE Security & Privacy* 99.5 (2003), pp. 35–43.

[165] Steven Padilla. *Android - Telnetd (Port 1035) with Parameters Shellcode (248 bytes)*. https://www.exploit-db.com/exploits/38194/.

[166] Joon S Park and Joseph Giordano. “Role-based profile analysis for scalable and accurate insider-anomaly detection”. In: *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*. IEEE. 2006, 7–pp.

[167] Keon Chul Park and Dong-Hee Shin. “Security assessment framework for IoT service”. In: *Telecommunication Systems* (2016), pp. 1–17. DOI: 10.1007/s11235-016-0168-0.

[168] Bryan Parno, Adrian Perrig, and Virgil Gligor. “Distributed Detection of Node Replication Attacks in Sensor Networks”. In: *2005 IEEE Symposium on Security and Privacy (S&P’05)*. IEEE. 2005, pp. 49–63. DOI: 10.1109/sp.2005.8.

[169] Roberto Perdisci et al. "McPAD: A multiple classifier system for accurate payload-based anomaly detection".
In: *Computer Networks* 53.6 (2009), pp. 864–881.

[170] Nicola Pergola et al.
"Time domain analysis of robust satellite techniques (RST) for near real-time monitoring of active volcanoes and thermal precursor identification".
In: *Physics and Chemistry of the Earth, Parts A/B/C* 34.6 (2009), pp. 380–385.

[171] Pavlos Protopapas et al.
"Finding outlier light curves in catalogues of periodic variable stars".
In: *Monthly Notices of the Royal Astronomical Society* 369.2 (2006), pp. 677–696.

[172] J Ross Quinlan. "Generating production rules from decision trees." In: *ijcai*. Vol. 87. 1987, pp. 304–307.

[173] Malek Ben Salem, Shlomo Hershkop, and Salvatore J Stolfo.
"A survey of insider attack detection research".
In: *Insider Attack and Cyber Security* (2008), pp. 69–90.

[174] Arthur L Samuel.
"Some studies in machine learning using the game of checkers".
In: *IBM Journal of research and development* 44.1.2 (2000), pp. 206–226.

[175] Stephen Schwab et al.
"SEER: A security experimentation environment for DETER".
In: *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*. USENIX Association. 2007.

[176] Risk Based Security. *2016 Data Breach Trends – Year In Review*. Tech. rep.

[177] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj.
"OpenStack: Toward an Open-source Solution for Cloud Computing".
In: *International Journal of Computer Applications* 55.3 (2012), pp. 38–42.
DOI: 10.5120/8738-2991.

[178] Yuru Shao, Xiapu Luo, and Chenxiong Qian.
"RootGuard: Protecting Rooted Android Phones".
In: *Computer* 47.6 (2014), pp. 32–40. DOI: 10.1109/mc.2014.163.

[179] Christos Siaterlis, Andres Perez Garcia, and Bela Genge.
"On the Use of Emulab Testbeds for Scientifically Rigorous Experiments".
In: *IEEE Communications Surveys & Tutorials* 15.2 (2013), pp. 929–942.
DOI: 10.1109/surv.2012.0601112.00185.

[180] The Snort Project. *Snort® Users Manual*. Comp. software. Cisco, 2016.

[181] Somerset Recon. *Hello Barbie Security: Part 2 - Analysis*.
http://www.somersetrecon.com/blog/2016/1/21/hello-barbie-security-part-2-analysis. Jan. 2016.

[182] Yingbo Song, Angelos D Keromytis, and Salvatore Stolfo. "Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic".
In: *Network and Distributed System Security Symposium 2009: February 8-11, 2009, San Diego, California: Proceedings*. Internet Society. 2009, pp. 121–135.

[183] Eduardo J Spinosa et al.
"Novelty detection with application to data streams".
In: *Intelligent Data Analysis* 13.3 (2009), pp. 405–422.

[184] William Stallings. *Cryptography and network security: principles and practices*. Prentice Hall, 2005.

[185] Stuart Staniford, James A Hoagland, and Joseph M McAlerney. "Practical automated detection of stealthy portscans". In: *Journal of Computer Security* 10.1, 2 (2002), pp. 105–136.

[186] Andreas Stolcke. "Entropy-based pruning of backoff language models". In: *arXiv preprint cs/0006025* (2000).

[187] Pedro Strecht, João Mendes-Moreira, and Carlos Soares. "Merging Decision Trees: a case study in predicting student performance". In: *International Conference on Advanced Data Mining and Applications*. Springer. 2014, pp. 535–548.

[188] Dafydd Stuttard and Marcus Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. John Wiley & Sons, 2011.

[189] Symantec. *An Internet of Things Reference Architecture*. White paper. 2016. URL: https://www.symantec.com/content/dam/symantec/docs/white-papers/iot-security-reference-architecture-en.pdf.

[190] David Martinus Johannes Tax. "One-class classification". In: (2001).

[191] David MJ Tax and Robert PW Duin. "Data domain description using support vectors." In: *ESANN*. Vol. 99. 1999, pp. 251–256.

[192] David MJ Tax and Robert PW Duin. "Support vector domain description". In: *Pattern recognition letters* 20.11 (1999), pp. 1191–1199.

[193] Choon H Teo et al. "Convex learning with invariances". In: *Advances in neural information processing systems*. 2007, pp. 1489–1496.

[194] *The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things*. https://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf. EMC Digital Universe with Research & Analysis by IDC, 2014.

[195] *The GNU Netcat project*. http://netcat.sourceforge.net/.

[196] Alvarez Torrano-Gimenez Perez-Villegas. *HTTP dataset CSIC*. http://www.isi.csic.es/dataset/. 2010.

[197] TrendLabs. *Hazards Ahead Current Vulnerabilities Prelude Impending Attacks*. Tech. rep.

[198] TrendLabs. *The Reign of Ransomware*. Tech. rep.

[199] Alexey Tsymbal. "The problem of concept drift: definitions and related work". In: *Computer Science Department, Trinity College Dublin* 106 (2004).

[200] Daniel Vecchiato, Marco Vieira, and Eliane Martins. "The Perils of Android Security Configuration". In: *Computer* 49.6 (2016), pp. 15–21. DOI: 10.1109/mc.2016.184.

[201] Haixun Wang et al.
"Mining concept-drifting data streams using ensemble classifiers".
In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'03*. ACM. Washington, D.C.: Association for Computing Machinery (ACM), Aug. 2003, pp. 226–235.
DOI: 10.1145/956750.956778.

[202] Juan Wang, Qiren Yang, and Dasen Ren.
"An intrusion detection algorithm based on decision tree technology".
In: *IEEE Asia-Pacific Conf. on Information Processing (APCIP)*. Vol. 2. 2009, pp. 333–335.

[203] Ke Wang, Gabriela Cretu, and Salvatore J. Stolfo.
"Anomalous payload-based worm detection and signature generation".
In: *International Workshop on Recent Advances in Intrusion Detection*. Vol. 3858. LNCS. Seattle, WA, USA: Springer, 2005, pp. 227–246.
DOI: 10.1007/11663812_12.

[204] Ke Wang, Gabriela F. Cretu, and Salvatore J. Stolfo.
"Anomalous Payload-Based Worm Detection and Signature Generation".
In: *Int. Symp. on Recent Advances in Intrusion Detection (RAID)*. 2005, pp. 227–246.

[205] Ke Wang, Janak J Parekh, and Salvatore J Stolfo.
"Anagram: A content anomaly detector resistant to mimicry attack".
In: *Recent Advances in Intrusion Detection*. Springer. 2006, pp. 226–248.

[206] Edward WD Whittaker and Bhiksha Raj.
"Quantization-based language model compression." In: *INTERSPEECH*. 2001, pp. 33–36.

[207] Gerhard Widmer and Miroslav Kubat.
"Learning in the presence of concept drift and hidden contexts".
In: *Machine learning* 23.1 (1996), pp. 69–101.

[208] Stefan Windmann et al. "A stochastic method for the detection of anomalous energy consumption in hybrid industrial systems".
In: *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*. Bochum, Germany: IEEE, 2013, pp. 194–199.
DOI: 10.1109/indin.2013.6622881.

[209] Petra Witschel et al. "POS-Based Language Models for Large Vocabulary Speech Recognition on Embedded Systems".
In: *Ninth European Conference on Speech Communication and Technology*. 2005.

[210] Michael Woolridge and Michael J. Wooldridge.
Introduction to Multiagent Systems.
New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN: 047149691X.

[211] Robert Wright et al.
"MODVOLC: near-real-time thermal monitoring of global volcanism".
In: *Journal of Volcanology and Geothermal Research* 135.1 (2004), pp. 29–49.

[212] Cheng Xiang, Png Chin Yong, and Lim Swee Meng.
"Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees".
In: *Pattern Recognition Letters* 29.7 (2008), pp. 918–924.

[213] Sun Yan-qiang and Wang Xiao-dong.
"Jamming Attacks and Countermeasures in Wireless Sensor Networks".
In: *From Principle to Practice*. IGI Global, 2010, pp. 334–352.
DOI: 10.4018/978-1-61520-701-5.ch015.

[214] Thiago Zaninotti. *Unfiltered Header Injection in Apache 1.3.34/2.0.57/2.2.1*.
http://www.securityfocus.com/archive/1/433280. 2006.

[215] Bruno Bogaz Zarpelão et al.
"A survey of intrusion detection in Internet of Things".
In: *Journal of Network and Computer Applications* 84 (2017), pp. 25–37.

[216] Jiong Zhang, Mohammad Zulkernine, and Anwar Haque.
"Random-forests-based network intrusion detection systems".
In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.5 (2008), pp. 649–659.

[217] Yan Zhou et al. "Adversarial support vector machine learning".
In: *ACM SIGKDD Int. Conf. on Knowledge discovery and Data Mining*. 2012, pp. 1059–1067.

[218] Dekun Zou et al. "Steganalysis based on Markov model of thresholded prediction-error image".
In: *2006 IEEE International Conference on Multimedia and Expo*.
Toronto, ON, Canada: IEEE, 2006, pp. 1365–1368.
DOI: 10.1109/icme.2006.262792.