UNIVERSITÀ
DI PAVIA

UNIVERSITY OF PAVIA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

PhD in Electronics, Computer Science
and Electrical Engineering
XXXIII Cycle

PHD THESIS

AI-aware Network Security Assessment: A Graph based Approach

Candidate: Giuseppe Nebbione

Supervisor: Prof. Maria Carla Calzarossa

A.A. 2019/2020

UNIVERSITY OF PAVIA

DOCTORAL THESIS

AI-aware Network Security Assessment: A Graph based Approach

Author:
Giuseppe NEBBIONE

Supervisor:
Prof. Maria Carla CALZAROSSA

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

in the

Performance Evaluation Laboratory
Department of Electrical, Computer and Biomedical Engineering

August 6, 2021

Declaration of Authorship

I, Giuseppe NEBBIONE, declare that this thesis titled, "AI-aware Network Security Assessment: A Graph based Approach" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITY OF PAVIA

Abstract

Faculty of Engineering
Department of Electrical, Computer and Biomedical Engineering

Doctor of Philosophy

**AI-aware Network Security Assessment:
A Graph based Approach**

by Giuseppe NEBBIONE

The number of cyber threats continues to evolve at a rapid pace, thus leading to an increasing number of security incidents. Hence, it is of paramount importance to properly secure digital infrastructures and services by assessing the potential security risks. For this purpose, over time, assessment procedures have been developed and published in the form of industry standards (e.g., ISO27001, OWASP, NIST P-800-42). Unfortunately, these procedures are far from being fully automated and integrated with AI technologies. To fill this gap, this thesis work aims at designing and developing a methodological framework and a toolset relying on the combined application of AI and graph-based algorithms. These AI-aware security assessment methodology, techniques and tools advance the state of the art in that they strongly enhance security assessment approaches. More precisely, graph based techniques are able to reveal non-obvious properties and relationships between the various components of a technological infrastructure. This information can be processed in order to extract useful features that, when used to train an AI classifier, allows to determine whether the target infrastructure is vulnerable.

As a result of this work, an AI-aware network security assessment toolset has been designed and developed in the domain of enterprise networks (i.e., complex computer networks based on directory services). This toolset relies on graph shortest paths and implements state of the art machine learning and statistical modeling techniques to determine critical points within an infrastructure and automatically detect vulnerabilities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The exponential growth of devices connected to Internet has led to a significant increase of cyber attacks which cause disastrous consequences and big damages to the companies and organizations being attacked. The number of cyber threats continues to evolve at a rapid pace, with an increasing number of data breaches and new families of malware developed each year.

The attackers behind these threats are driven by different motivations and goals. In fact, attacks are nowadays carried out by different actors who are commonly identified as: cybercriminals, generally driven by monetary purposes; hacktivists, generally driven by activism; state sponsored actors, generally driven by defensive or offensive cyberwarfare operations. As a matter of fact, the United States Department of Defense recognizes the use of computers and the Internet to conduct warfare in cyberspace as a threat to national security, but also as a platform for attack [1].

The goal of attackers is not always the one of compromising the target systems. They might also aim at gathering data to be used to perform intelligence operations and get insights on the target. This is especially the case in industrial and state-sponsored espionage. Hence, it is also important for a company to be constantly aware of the information being exposed and how this information could be abused to harm the company itself.

It has been reported that over the past years, organizations have been facing growing challenges in computer security. In fact, while they increased the annual spending on cybersecurity infrastructures, the trend of attacks has continued to escalate at unprecedented speed [2].

In this landscape, one of the primary choices used by the attackers is malware, which acts as a weapon to carry out malicious intents in the cyberspace. As pointed out by Statista [3] the number of new malware developed by attackers follows an exponential growth and is increasing every year. Attackers do not limit themselves to simple attacks but in many cases are organized and able to engineer novel attacks to achieve their purpose.

Malware is often deployed by e-mail through the use of phishing and spear phishing techniques which in this context are the old time favorites by attackers. Moreover, besides being the most used vector, compromised

[1] https://www.businessinsider.com/us-military-cyberwar-2016-5

[2] https://www.fireeye.com/content/dam/fireeye-www/virtualsummit/pdfs/cyber-trends-2020-report.pdf

[3] https://www.statista.com/statistics/680953/global-malware-volume/

e-mail accounts are also a very fruitful achievement in terms of monetary costs.

The Federal Bureau of Investigation Internet Crime Complaint Center reported that 26.2 billion US Dollars was the amount of international losses reported due to mail compromise between June 2016 and July 2019 [4].

Another recurring issue for what concerns the attacks is related to Distributed Denial of Service attacks. As documented by Akamai [5], between July 2019 and June 2020, the commerce category faced 125 DDoS attacks, 90% of these attacks were against organizations operating in the retail sector, while the remaining ones were targeted toward touristic services. These attacks may happen as a form of hacktivism, protest or in general to disrupt specific services.

Another common issue in the cybersecurity attack landscape is related to data breaches which are mostly affecting the healthcare and government sectors. There was an 80% increase in data breaches in the health sector from 2017 to 2019 [6].

In addition, the COVID-19 pandemic increased the smart-work activities and led companies to expose services to their employees, thus increasing their attack surface. Cybercriminals have quickly coped with this situation, utilizing it as an opportunity to launch attacks. In fact, as reported by McAfee [7], there was recently an increase in malware attacks targeting the public, healthcare and education sectors.

Over the years, proactive and reactive procedures have been developed to assess the security of implementations and services. These procedures have been organized and published in the form of industry standards (e.g., ISO27001, OWASP, NIST P-800-42). Unfortunately, they are far from being fully automated and they still do not significantly exploit AI technologies in contexts where it would be advantageous with respect to the usage of heuristics.

Hence, to cope with the evolving threat landscape, it is of paramount importance to properly secure digital infrastructure and services by assessing security risks associated with each asset.

To secure infrastructures, a multitude of tools has been developed. In particular, in recent years there has been a growing interest in AI technologies and also in AI-powered cybersecurity. In fact, machine learning models can provide insights about complex patterns found in advanced attack scenarios.

For this reason, it becomes particularly challenging to combine AI and cybersecurity and explore the possibilities opened by the application of machine learning techniques to the computer security field.

This thesis work addresses these challenges by modeling computer networks as graphs and taking advantage of machine learning techniques to

[4] https://www.ic3.gov/Media/Y2019/PSA190910

[5] https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-security-loyalty-for-sale-retail-and-hospitality-fraud-report-2020.pdf

[6] https://www.statista.com/statistics/798564/number-of-us-residents-affected-by-data-breaches/

[7] https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-july-2020.pdf

gain better insights on an infrastructure and its corresponding weaknesses. More precisely, a methodological framework and a toolset for performing AI-based security assessment within complex infrastructures are proposed.

The layout of this research work is as follows. Chapter 2 presents a general overview of the proposed approach by describing the motivation and reasons behind it and a high level description of the proposed framework. Chapter 3 describes common security issues and typical countermeasures implemented to tackle these issues. Chapter 4 discusses the state of the art regarding computer security and in particular literature combining artificial intelligence based techniques with security. Machine learning concepts and graph theory notions are briefly highlighted in Chapter 5. The technologies behind enterprise networks, Active Directory and the corresponding security issues are discussed in Chapter 6. The proposed methodological framework is described in Chapter 7, while details on the toolset implementing this framework are given in Chapter 8. Chapter 9 discusses some experimental results observed in artificially generated test environments developed to assess the benefits of the proposed framework. Finally, Chapter 10 presents some conclusions and possible developments of this research work.

Chapter 2

Background

There is an urgent need in outperforming attackers by developing and updating proactive and reactive measures to secure networks. To achieve this goal, this thesis work focuses on proactive assessment of advanced attacks within complex computer environments. In particular, the work aims at designing and developing a methodological framework and a toolset based on the combined application of graph algorithms and artificial intelligence techniques. This comprehensive framework will bridge the gap between AI and computer network security and will enhance the state of the art in the area of security assessment procedures.

The focus of this research is security assessment in the domain of enterprise networks, i.e., complex computer networks relying on a directory service, such as Microsoft Active Directory. In fact, enterprise networks are particularly complicated to manage, because they are characterized by the combination of many diverse devices, technologies and protocols, whose actual behavior might be difficult to understand, thus leaving possible points of access for attackers. As a consequence, this complexity might lead to a huge attack surface for advanced threat actors. The choice of analyzing Active Directory environments is justified by the fact that these environments although do not represent the totality of enterprise systems they are prolific enough to be used by about ninety percent of Fortune 500 companies [1].

In this thesis work, these complex scenarios are modeled through graphs, whose nodes represent network entities and whose edges represent the relations between entities. The graphs considered in this work, also known as "attack graphs", represent the set of all paths that can be used by an attacker to compromise a network by obtaining the role of the network administrator. The concept of attack graph is well known in computer security although these graphs are often customized to specific scenarios or technologies. In fact, there is a lack of formalisms and tools to represent graphs able to model general computer networks and scenarios. To cope with this issue, the proposed framework focuses on Active Directory environment and is general enough to be applied to a big variety of scenarios.

Applying graph theory to computer networks has many benefits. In particular, graph models allow the discovery of shortest paths in terms of weaknesses in a network, that is, the most easily exploitable path for an attacker, or the path in which it may be easier to locate vulnerabilities. In this context, the evaluation of the relationships between network entities and paths within a graph is more effective than the isolated evaluation of vulnerabilities

of a single entity. More precisely, the evaluation of single endpoints may not reveal the complex attack scenarios resulting from the combination of multiple weaknesses. The relationships inside the graph can easily reveal these vulnerabilities.

Although some works have investigated the application of graph models to computer security, the combined application of graph theory and AI to complex infrastructures has not been yet studied. For this reason, this work proposes a framework that relies on the use of graphs integrated with machine learning supervised techniques. An overview of this framework is depicted in Figure 2.1.

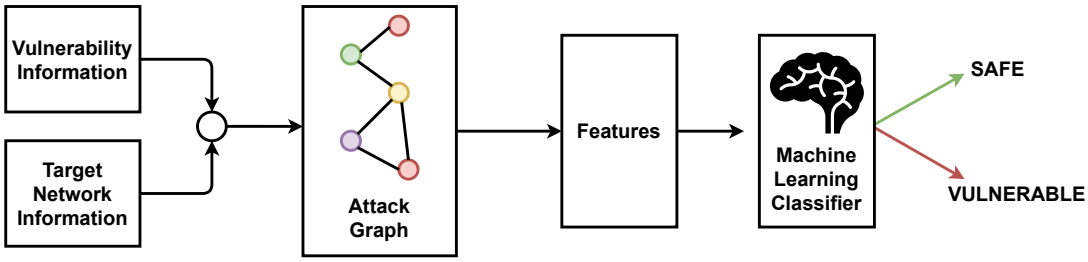


FIGURE 2.1: Pipeline representing the main steps of the proposed framework

As can be seen, an attack graph is built using information about the network to be assessed and the priorities associated with potential vulnerabilities. For example, information about the network can be retrieved through the use of active and passive scanners. The analysis of the attack graph provides insights about the network topology and the paths that may be at risk and allows to derive features for training the classification model. A vulnerability score is associated with each path to determine the severity of the various weaknesses being identified. As a result, the classification process will assess whether the network is vulnerable or safe. This combined automatic application of machine learning and graph theory is novel in that it has the potential to enhance the effectiveness of assessment procedures.

The toolset developed as a result of this research activity relies on a graph database used to store, retrieve and analyze node relationships. In addition, the toolset implements state of the art machine learning and statistical modeling techniques used to perform classification tasks based on the information provided by the graph nodes and edges. The design of the toolset also involved the development of passive and active scanners that can be used for discovery purposes on Active Directory networks as well as the construction of a generator of artificial complex environments that can be used for testing and research purposes to cope with the lack of real data.

Chapter 3

Security Issues

Even though large scale attacks cause big damages, small scale attacks can be equally dangerous since they often go undetected for quite a long time. Therefore, it is compelling to strengthen cybersecurity by identifying what needs to be secured and by developing countermeasures that take account of the specific characteristics and physical limitations of individual devices.

3.1 Introduction to Security Issues

Computer security is defined as "the branch of science that measures, controls and ensures confidentiality, integrity, and availability of information system assets including hardware, software, firmware, and information being processed, stored, and communicated" [2]. Keeping our information and digital infrastructures secure in a world where digitalization is happening at an ever growing rate is becoming more and more important. In fact, nowadays computers are an ubiquitous technology that, although allow us to increase productivity and access a huge amount of information within seconds, also carry a significant amount of threats to their users. For this reason, there is an increasing need for companies and citizens to protect their digital assets.

Protecting assets means in general to have guarantees on confidentiality, integrity and availability (the so called "CIA triad"). A cyberattack always affects at least one of these three elements. In detail, confidentiality is a necessary component of privacy and refers to the ability to protect data and infrastructures from accesses that are not authorized. Integrity refers to the ability to prevent our data and systems from being changed in an unauthorized manner, while availability refers to the ability to access data, systems or services when needed.

The problem of making systems immune to attacks is particularly difficult since security and productivity are often contrasting concepts. In addition, being able to point out exactly when we are sufficiently secure in a certain context is a very difficult task.

Cyberattacks are conducted by taking advantage of vulnerabilities present in information systems. In order to be able to analyze attacks and understand mitigations, security threats must be understood and analyzed.

A threat in this context represents something that has the potential to cause harm to digital assets. Threats tend to be related to environments. For example, a malicious software for the Microsoft Windows operating system

represents a threat for Windows machine, while it does not represent a threat for a machine running any other operating system.

In this chapter, the actors behind these threats and the main sources of threats are explored by analyzing common security issues. Additionally some of the ways in which organizations are fighting these threats nowadays are described.

3.2 Threat actors

A threat actor or malicious actor is a person or entity responsible for an event or incident that impacts, or has the potential to impact, the safety or security of another entity [3]. Most often, the term is used to describe individuals and groups that perform malicious acts against organizations of various types and sizes.

Threat actors can be categorized as either intentional or unintentional and internal or external. A threat actor is considered "intentional" if it explicitly has intentions to cause harmful events or damage. On the other hand, an "unintentional" threat actor is unaware of representing a threat. Moreover threat actors can be categorized into: "internal" if it belongs to the targeted organization or "external" if it is external to the targeted organization.

Threat actors have different purposes and background. Common types of actors involved in threats include but are not limited to malicious insiders, cyber criminals, cyber-terrorists, nation-states, hacktivists.

Malicious insiders are the most frequent and successful threat in small companies. According to the Ponemon Institute, the average cost related to insider-related incidents was around $\$513,000$ only in 2018, even though insider-related incidents can cost a company up to $\$8.76$ million a year.

In addition, the percentage of companies that suffered from malicious insiders is constantly increasing as shown by Ekran System[1].

3.3 Security threats

When talking about security threats, the concepts of risk and vulnerability are of paramount importance. Risk is the likelihood that something bad will happen. For a risk to materialize in a particular environment, it is necessary to have both a threat and a point in the environment vulnerable to that specific threat (i.e., vulnerability) that the specific threat can exploit.

To make an example, if we live in a neighborhood with many thieves (threat), and our house does not have a door (vulnerability), we most definitely have a risk.

To deal with threats, security analysts need methods to manage, classify and prioritize security vulnerabilities. For this purpose, MITRE[2], a not-for-profit organization provides information about vulnerabilities in the form of

[1] https://www.ekransystem.com/en/blog/insider-threat-statistics-facts-and-figures
[2] https://www.mitre.org/

a database composed by a list of records, each containing an identification number, a description, and at least one public reference for a publicly known vulnerability. Each record associated with a vulnerability is known as Common Vulnerability Exposure (CVE). This public database represents a common knowledge base for publicly disclosed vulnerabilities in the computer security industry.

In addition, the National Institute of Standards and Technology (NIST)[3] periodically analyzes the database provided by MITRE and assigns a severity score to each CVE, ranging from 1 to 10, known as Common Vulnerability Scoring System, (CVSS).

In the next sections some of the most common threats used by attackers are detailed.

3.3.1 Denial of Service

A Denial-of-Service (DoS) attack is a cyber-attack in which the attacker seeks to make machines or network resources unavailable to their intended users by temporarily or indefinitely disrupting the services being offered. Denial of service is typically accomplished by flooding the targets with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled [4].

In general, nowadays this attack is mostly performed in a variant denoted as Distributed Denial of Service (DDoS), because the incoming traffic originates from many different sources. Another characteristic of DoS or DDoS is that it is impossible to stop the attack simply by blocking a single source of the incoming traffic.

Frequently motivations behind DoS attacks are revenge, blackmail or activism. Some vendors provide the so called “booters” or “stressers” which are often misused by criminals as platforms to conduct DoS attacks. Modern DDoS attacks take advantage of botnets that frequently include compromised IoT devices. A report from A10 Networks[4] indicated that the company tracked more than 20.3 million infected devices, including IoT devices, and servers taking part to a DDoS attack.

3.3.2 Malware

Malicious software, or malware, is any software that causes harm to a user, computer, or network. A malware plays a key role in most computer intrusion and incidents [5]. Malware comes in different forms, and one of the ways to categorize malicious software refers to the actions it performs. In this context, the main types of malware are classified into the following categories:

- Virus: this type of malware modifies legitimate files in such a way that their execution causes the execution of some unwanted instructions;

[3] https://www.nist.gov

[4] https://www.a10networks.com/marketing-comms/reports/state-ddos-weapons/

- Worm: this is a self-replicating malware that spreads without any end-user action. For this reason, worms are generally devastating for enterprises;
- Trojan: this is a malware masquerading as a legitimate program but containing malicious instructions;
- Ransomware: this is a malware whose goal is to "lock" the target machine and requires a ransom to unlock the system;
- Adware: this is a malware whose goal is to expose users to unwanted or potentially malicious advertising;
- Spyware: this is a malware whose goal is to collect and send data to a third party.

It is important to remark that a malware often spans different categories, depending on the actions being performed.

Another common classification used to characterize a malware is based on its target. In this context, a distinction can be made between mass malware, that is malware designed to affect as many machines as possible, and targeted malware, that is a one-of-a-kind software tailored to a specific organization and scenario.

The number of new malware developed is increasing every year and according to the ENISA Threat Report[5] in 2019, about 10.1 billion of Euros have been paid by organizations in ransoms with 45% of the targeted organizations paying the ransom.

3.3.3 Phishing

Phishing is an attack where the attacker exploits social engineering techniques and performs identity spoofing to trick a target victim [6]. Phishing traditionally relies on forged e-mail, SMS messages and other means, often mimicking an online bank or a system administrator in an organization asking the user to install some software. In particular, Phishing emails provide a common mean to infiltrate computer systems of organisations by encouraging employees to enter login credentials on a specially crafted fake website, click on malicious links or attachments. Although this attack is typically carried out by means of email messages, instant messaging software, or text messaging (smishing) represent emerging vectors.

Phishing represents the most common attack vector used by cybercriminals. In fact, phishing emails are used either to guide the user to a bogus website with the goal of stealing its credentials or guide the user in downloading a malware in the form of a fake Office document or software update to execute.

Phishing attacks can be categorized depending on their target:

- Bulk Phishing: mass-phishing attacks without any specific target;

[5] https://www.enisa.europa.eu/publications/ransomware

- Spear Phishing: phishing attacks directed towards specific individuals or companies.

Phishing attacks can be simple or very complex. In fact, these attacks can be used in combination with domain squatting techniques, which consist in the registration of domain names similar to the original ones to be used in the phishing attack with the intent of increasing the probability of fooling the targets.

Phishing is so common that, as reported by KeepNetLabs, over 60,000 phishing websites have been reported in March 2020 and 83% of the detected attacks for spear phishing use brand impersonation to trick the targets[6].

3.3.4 Data Breach

A data breach is the intentional or inadvertent exposure of confidential information to unauthorized parties [7]. Data breaches hurt businesses and consumers in a variety of ways since as technology progresses, more and more of our information is being digitalized. As a result, cyberattacks attempting to steal personal data have become significantly more interesting and profitable for criminals. Cybercriminals exploit personally identifiable information (PII) to steal money, compromise identities, or sell data over dark markets.

Data breaches can occur for a number of reasons, including unintentional data breaches. Targeted data breaches are typically carried out thanks to the following problems:

- System vulnerabilities: out-of-date software represents a vulnerability that, under certain conditions, may allow attackers to compromise systems and steal data;

- Weak passwords: weak and insecure user passwords are easier for hackers to guess or to bruteforce. Being able to compromise an administrative account can be still rewarding since provides the attacker with additional privileges or information;

- Targeted malware attacks: attackers use social engineering techniques such as phishing or malware deployment to trick users into revealing their credentials, downloading malware attachments, or directing them to vulnerable websites. These attacks may reveal sensitive information about the target.

Corporations and businesses are extremely attractive targets for cybercriminals, simply due to the large amount of data that can be stolen within single successful attack.

Data breaches happen quite often and often cause big economic losses. In addition a data breach causes reputation damages to the affected company that can lead to bankruptcy.

[6]https://www.keepnetlabs.com/phishing-statistics-you-need-to-know-to-protect-your-organization/

According to statistics provided by IBM[7] the most affected sector by data breach is the healthcare with an average loss of $7.13 millions per data breach. On the other hand, the average total cost for companies due to data breaches is $3.86 million. IBM also reports that the average time passed to contain a breach is about 280 days.

3.3.5 Advanced Persistent Threat

An Advanced Persistent Threat (APT) is a stealthy, well funded, organized group that is typically backed by a government or by important commercial entities. The term is used to describe advanced adversaries that are focused on critical data with the goal of exploiting information in a covert manner [8]. Since APT actors are highly skilled, sometimes they can bypass also nontrivial security mitigations. In fact, attacks originating from APTs are generally infiltrating target networks by taking advantage of zero days, that are vulnerabilities not publicly disclosed.

APTs, as described, are in general operated by nation states or state sponsored groups, with computer security experts denoted as "APT operators". Generally operators behind APTs have at their disposal a full spectrum of intelligence gathering techniques and zero day exploits to use. In fact, in these scenarios, operators have very specific objectives which often include but may not be limited to intelligence gathering. As the name suggests, these attacks are very complex and difficult to detect.

A famous example of APT was the Stuxnet computer worm, which targeted the computer systems of Iran's nuclear program. In this specific case, the Iranian government considered the Stuxnet creators to be an advanced persistent threat.

3.4 Defense Mechanisms

In order to mitigate and tackle cybersecurity attacks, organizations have started to integrate in their policies defensive measures. These measures vary in nature and can range from a simple vulnerability scan to a full fledged long term adversary simulation, where a targeted real-world like attack is emulated using a group of security specialists.

In general, we can distinguish between reactive and proactive defensive measures. Often both should be adopted to be protected from cyberattacks. Reactive methods include the set of techniques and tools to be used whenever an attack is taking place or has already happened. Their goal is to reduce the likelihood associated to the success of the attack and to quickly remediate the incident. Examples of reactive tools and techniques include general incident response plans or antivirus software used to limit the spread of an attack.

On the other hand, proactive methods seek to check the security of a system without the necessary presence of an incident. These methods can range

[7] https://www.ibm.com/security/data-breach

from simple vulnerability assessments to penetration testing activities or to red team operations aimed at testing all the security controls. In general proactive methods should be performed periodically, since they provide a snapshot of the security status of the infrastructure at a certain time.

One of the main proactive methods used in industry is penetration testing. Penetration testing activities are conducted by security specialists called "penetration testers" or in these contexts simply "testers". Testers are in general guided by a standard methodology following different steps to assess the security of a system. There are different penetration testing standards, some of them are general while others are focused to specific technologies (e.g., web application testing, IoT testing).

In the following sections, the incident response and the penetration test methodologies are briefly described.

3.4.1 Incident Response

The incident response plan is a list of steps to be taken in case of incident and included in a methodology an organization uses to respond to and handle cyberattacks. An incident response aims at reducing the damage that a computer incident may bring to customers, intellectual property, resources and brand value. In terms of enterprise security, protection of assets is particularly difficult especially because of the large attack surface. In these contexts, attackers have to succeed only once, while system administrators defending the infrastructure cannot make mistakes. Thus, companies not only have to constantly protect their assets but it is equally important to design and manage appropriate incident response plans.

Investigation is also a key component in order to learn from the attack and better prepare for future threats. Because many companies today usually experience a breach, a well-developed and repeatable incident response plan is the best way to protect the company assets.

The team responsible for Incident Response management is known as Computer Incident Response Team (CIRT). This team generally includes managers, security analysts and auditors. Note that in the context of incident response, any cyber attack is considered a network "breach".

According to the SANS Institute[8] the six key steps included in a response plan as shown in Figure 3.1, are:

1. Preparation: this phase deals with the development of policies and procedures to follow in case of a cyber breach. It includes determining the composition of the response team and the triggers that will alert internal mechanisms. Key to this process is effective training to timely respond to a breach and documentation that is used to record actions taken for later review;

2. Identification: this phase deals with the detection of the breach and the enabling of a quick response. A team of security analysts identifies breaches using a threat intelligence platform, intrusion detection

[8]https://www.sans.edu

FIGURE 3.1: Incident Response steps according to SANS Institute

systems, and firewall logs. A fundamental step in this phase is represented by "Threat intelligence", that is, the analysis of current threat trends and common tactics used by adversaries fundamental part in this step;

3. Containment: this phase deals with the containment of the damage and the prevention of further penetration after having identified a computer incident. This can be accomplished by taking specific parts of the infrastructure offline and relying on backups to maintain operations;

4. Eradication: this phase deals with the neutralization of the threat and restoration of internal infrastructure and services to their previous state. This step may involve additional monitoring activities to ensure that affected systems are no longer vulnerable to further attacks;

5. Recovery: this phase deals with the validation that all affected systems are no longer compromised and can return to normal working condition. This also requires setting timelines to fully restore operations and continued monitoring for any abnormal network activities. At this step, it is possible to estimate the costs of the breach and the corresponding damage;

6. Lessons Learned: the most important stage of the incident response plan is related the final steps of the methodology, where the incident response team members meet to determine what went wrong and how to improve the methodologies for future breaches. This can involve the evaluation of the current policies and procedures, as well as the analysis of specific decisions the team made during the incident. A final analysis of the incident should be documented into a report and used for future training.

As already mentioned, incident response plans and reactive techniques in general are not sufficient to guarantee the security of a digital infrastructure. For this purpose, we need to couple reactive strategies with proactive techniques. In the next section, the main proactive technique, that is penetration testing, is detailed.

3.4.2 Penetration Testing

A penetration test, also known as a pen test, is a simulated cyber attack against a computer infrastructure to check for exploitable vulnerabilities. Penetration testing activities are conducted by specialized security professionals known as "penetration testers" or simply as "testers". Since digital infrastructures include technologies that can be very different, such as web applications, IoT devices or full-fledged Active Directory environments, different penetration testing methodologies exist. In fact, penetration testers may be specialized in one or more areas of testing.

Penetration testing methodologies can be subdivided into general purpose methodologies, web application testing methodologies, network methodologies and wireless network methodologies.

In addition, penetration testing can be further categorized as follows:

- White box: full network and system information is shared with the tester. In the case of a network pen test this information may include network maps and credentials. White box approach is often chosen to reduce the time required to perform tests and reduce the resulting costs. Another advantage offered by this type of approach is that the testers can simulate an attack on the target system using more attack vectors;

- Black box: no information is shared with the tester. This condition has the advantage of being closer to a real world scenario where an attacker has little or no information about the target. These kinds of tests are more expensive and require longer time to be performed;

- Grey box: only a limited amount of information is shared with the tester. This is an hybrid approach that represents a trade-off in terms of costs and time between a white box approach and a black box approach.

In the case of network penetration testing, we can also make a distinction between internal and external penetration test procedures. An internal penetration test emulates the scenario of an attacker who has already access to a machine within the target network. On the other hand, an external penetration test emulates the scenario of an attacker that has no access to any target machine, thus, has to first gain a foothold into the target environment.

According to the Penetration Testing Execution Standard (PTES)[9], penetration testing activities as also shown in Figure 3.2

These steps are briefly detailed in what follows:

[9]http://www.pentest-standard.org

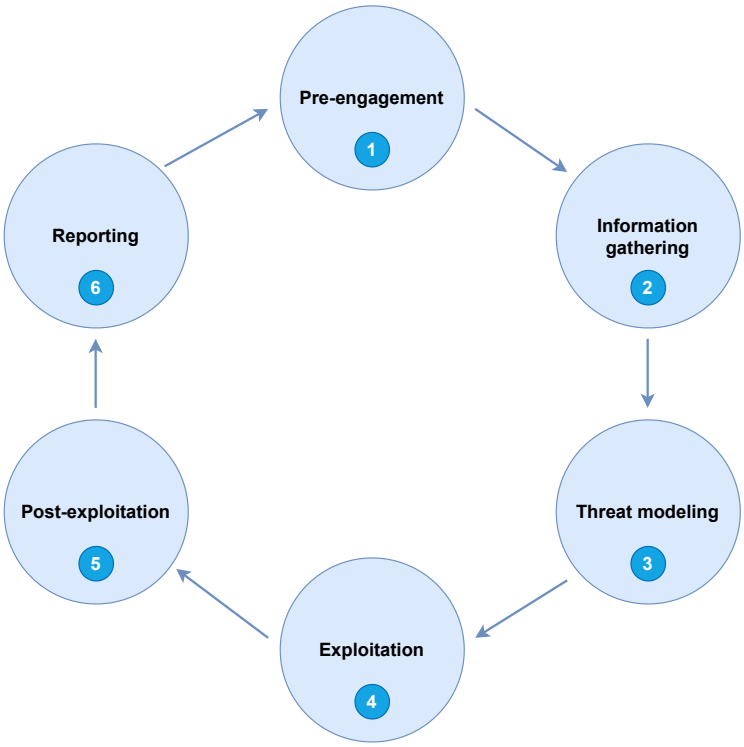


FIGURE 3.2: Penetration Testing steps according to PTES

1. Pre-engagement: the pre-engagement interactions represent the first step of every penetration test, when the type of test is outlined with the customer, and details about time, IP addresses and other information are agreed between the parts. This step is particularly important because the scope is defined and additional operational information may be provided to the testers. The scope of an engagement generally includes: IP addresses, subnet blocks, domain names or applications. The parties involved also establish the techniques the techniques allowed during the testing activities. In general for standard penetration testing activities for example, social engineering is not allowed, hence attackers cannot interact with employees of the target company. On the contrary, in the case of more extended assessments such as red team operations, social engineering techniques become a fundamental component and attackers may be allowed to trick employees, e.g., with phishing.

2. Intelligence Gathering: during this step a reconnaissance procedure against a target is performed to gather as much information as possible to be utilized when penetrating the target during the vulnerability assessment and exploitation phases. The more information gathered during this phase, the more attack vectors to be used in future phases. Intelligence gathering often is performed through Open Source Intelligence (OSINT) techniques. OSINT is a form of intelligence collection management that involves finding, selecting, and acquiring information from publicly available sources and analyzing it to produce actionable intelligence. This phase is particularly important, since it defines

the target attack surface. In this context, a larger attack surface may lead to a higher probability in finding vulnerabilities in the environment.

3. Threat modelling: this is the process aimed at identifying threats, such as structural vulnerabilities or the absence of appropriate security measures. In this phase the findings are enumerated and prioritized. Threat modeling can be used both by attackers and defenders. The purpose of threat modeling for defenders is to provide a systematic analysis of which security controls or measures need to be included, given the nature of the system, the probable attacker's profile, the most likely attack vectors, and the most desired assets by an attacker. On the other hand, for attackers, threat modelling defines possible attack scenarios or attack paths for compromising specific assets.

   At a high level a threat modeling process includes the following steps:

   - Gathering relevant documentation;
   - Identifying and categorizing primary and secondary assets;
   - Identifying and categorizing threats and threat communities;
   - Mapping threat communities against primary and secondary assets.

   Once the most important threats are modeled, attackers must plan which threat should be exploited.

4. Vulnerability Analysis: vulnerability testing is the process of discovering and enumerating flaws in systems and applications which can be later leveraged by an attacker in the exploitation phase. These flaws can include host and service misconfiguration, or insecure application design. The process used to find flaws varies and is highly dependent on the particular technology being tested. Vulnerability analysis can be active, when there is a direct interaction with the component being tested for security vulnerabilities, or passive, when an attacker leverages data found from third parties (e.g., search engines) or traffic analysis without directly interacting with the target to find possible flaws. The vulnerability analysis process involves the enumeration of the services running on the system and their version. In addition, it includes the usage of vulnerability scanners, that may provide useful insights on the vulnerabilities on the tested system. Unluckily, although vulnerability scanners are automatic they provide lots of false positives, that are, vulnerabilities detected by mistake. For this purpose, attackers should always verify each result produced by these automated tools.

5. Exploitation: the exploitation phase focuses solely on establishing access to a system or resource by bypassing security restrictions. The main focus of this phase is to identify the main entry point into the organization and high value target assets to compromise. Ultimately the attack vector should take into consideration the success probability

and highest impact on the organization. The techniques applied in this phase vary depending on the technology that is being tested.

6. Post-Exploitation: the purpose of this phase is to determine the value of the compromised system and to maintain control for later use. The value of a system is determined by the sensitivity of the data stored on it and its usefulness in further compromising the network. Post exploitation is a particularly critical step, because in most cases after the exploitation step attackers have the control of an unprivileged user and they generally need to elevate their privileges. In addition, attackers must also be able to install backdoors on the system to be able to maintain the access on the compromised host. Maintaining the access and escalating privileges are critical steps to further penetrate the network.

7. Reporting: the most important outcome of a penetration testing activity is the production of a report, where the tester communicates the outcome of the activity. The technical report details every step that was taken during the penetration testing exercise, outlining the steps that lead to the compromise of the target.

Let us remark that penetration testing is becoming an essential part of the entire systems and software development life cycle. Many organizations fully integrate assessment methodologies both for their applications and for their infrastructure. Nonetheless, being able to perform comprehensive assessment procedures on complex infrastructures is not an easy task. In fact, it is beneficial to be able to automate several aspects of penetration testing and integrate assessment tools with state of the art machine learning.

Chapter 4

State of the Art

In recent years, thanks to the massive digitalization and popularity of electronic systems, computing technologies pervade our society, that is becoming more and more interconnected. Vulnerabilities of devices are discovered with increasing frequency and their exploitation continues to accelerate. For these reasons, a large body of the literature has focused on analyzing and modeling computer security problems. In what follows, a brief review of the literature considering general computer security research directions is presented. In addition, works related to the research addressed by this thesis work are discussed.

In particular, the chapter reviews the use of graphs for modeling computer security issues and the software tools and frameworks developed.

4.1 Computer Security Research

Computer security is a broad topic whose applications are very diverse. It includes cryptography, computer networks and protocols security, threat modeling, web applications security, binary exploitation, to name a few. A large body of the literature investigates computer security in different scenarios, by also taking into account the evolution of the technological landscape.

The analysis of the literature has revealed that in the last few years there has been a growing interest both on theoretical topics focusing on new and emerging technologies, and on specific security technologies and problems applied to a variety of domains. From a theoretical perspective cryptography and threat modeling have been extensively investigated. In addition, security technologies have been frequently addressed in application domains such as, Internet of Things (IoT), cloud computing, and Software Defined Networking (SDN).

Cryptography is a relatively old and big field of research within computer security mainly focused on methods to obtain confidentiality in communications [9]. Many studies focus on efficient and robust techniques to encrypt data and improve confidentiality (see, e.g., [10, 11]). For example, in [10] lightweight cryptographic techniques are proposed in a direct communication scheme in a 5G IoT network. The communication scheme design is based on elliptic curve algorithms, which represent a fundamental tool for lightweight cryptography. In fact, nowadays there is a strong interest in

the implementation of efficient cryptographic techniques designed for low-powered devices because of the pervasive presence of integrated electronic systems, such as IoT devices, wireless sensors, SCADA instrumentation and novel low power CPU architecture technologies (see, e.g., [12–15]).

Another emerging research direction within cryptography is related to "quantum cryptography", that is, an interdisciplinary field whose aim is to take advantage of quantum mechanical properties to perform cryptographic tasks [16]. A significant number of recent research works explore the capabilities of quantum cryptography, its possible implementations and implications (see, e.g., [17–20]). For example, an entanglement-based secure quantum cryptography scheme that works over 1,120 km is proposed in [19]. This scheme relies on Quantum Key Distribution and significantly increases the secure distance (i.e., the maximum distance at which a secure communication can be established) by augmenting the overall security of quantum key distribution.

An important field of research within computer security transversal to the whole range of applications is threat modeling. Threat modeling is an abstract process by which potential threats, such as structural vulnerabilities or absence of appropriate safeguards, can be identified, enumerated, and corresponding mitigations can be prioritized. Different methodologies can be used to model threats. Several papers focused on the development of threat modeling techniques applied to different scenarios (see, e.g., [21–24]).

Recently, Novokhrestov et al. [25] proposed an approach to build threat models based on an actual computer network model. In particular, the paper distinguishes between information threats and system threats and presents a solution that overcomes the limitations provided by the subjective opinion of experts when compiling the list of threats.

Another research work [26] criticizes widely adopted threat modeling frameworks by illustrating their main shortcomings. The paper considers the current methodologies inadequate for the representation of different security concepts, data elements, abstraction levels, and deployment information. Hence, the paper supports the need of a dedicated, integrated language for threat modeling. The paper outlines that such a language is not readily available and that future candidate languages should consider a trade-off between the complexity of the language which makes adoption more challenging and its support for systematic and repeatable threat modeling.

Computer security literature has also addressed the IoT world. In fact, as shown in several research works (e.g., [27, 28]) security and privacy of many consumer IoT devices are characterized by some form of vulnerability. Given the number and wide variety of security issues, some studies focus on the identification and categorization of these issues. For example, Neshenko et al. [29] classify IoT-specific vulnerabilities into a multi-dimensional taxonomy designed to assist implementers in the development of new devices. Moreover, the research paper sheds light on corresponding technical details, consequences, mitigations and best practices.

As can be seen by a recent literature review by Aly et al. [30], IoT security and its applications have been extensively analyzed in the literature.

IoT security is particularly challenging because security features might be expensive in terms of power, while IoT devices have to function on low power. The interest in the improvement of the trade-off between security and power consumption led several research efforts. For example, Shamala et al. [31] provide an overview of lightweight cryptography algorithms developed for networks based on IoT devices. The analysis summarizes the current research challenges involved in IoT security. In detail, the analysis takes into account several lightweight cryptographic techniques in terms of memory consumption, execution time and code size and shows that the choice of the cryptographic technique depends on the requirements established by the application. The research paper takes into account the most popular performance metrics commonly considered for cryptographic algorithms and demonstrate that "Chaskey" is the best performing algorithm.

The efficiency of cryptographic techniques is not the only problem affecting IoT devices. In fact, misconfigurations caused by the plug-and-play nature of these devices are abused to perform different kinds of attacks. Zhou et al. [32] propose a set of features that uniquely characterize IoT devices, network subsystems and applications and discuss the potential threats and vulnerabilities associated with each feature as well as solutions and opportunities to tackle the threats.

Security in cloud environments is another hot topic investigated in the literature [33–35]. In fact, cloud infrastructures and related exploitation techniques are increasingly becoming popular and a successful attack could impact a significant number of organizations and users. Tabrizchi et al. [34] identify and categorize cloud environment security issues into five categories, namely: policies, users, data storage, applications and network. The paper also presents for each of these categories, the corresponding mitigations (that can be put in place by implementers). For example, one of the most important challenges in cloud computing security is represented by application vulnerabilities. In fact, cloud applications sometimes consist of millions of lines of code written by different programmers in different programming languages each with its vulnerabilities. Hence, software development best practices in terms of security should be strictly followed and software security should periodically be assessed.

A common issue that is transversal with respect to the categories outlined by Tabrizchi et al. [34] is related to DoS attacks. These attacks are very popular and may have catastrophic consequences in cloud environments. Saisindhutheja and Shyam [36] proposed a meta-heuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud environments. The research paper relies on an efficient technique that scans the feature space until the best occurring combination of features is identified.

In the framework of Software Defined Networking several solutions for ensuring security have been proposed in the literature. A survey of the state of the art outlining research works in this area is presented in [37]. This work identifies a taxonomy of SDN security issues is identified, this categorization highlights the main characteristics and contributions of many recent works. Hu et al. [38] develop a framework to facilitate the detection and resolution of

firewall policy violations in SDN networks based on OpenFlow. The framework checks network flow paths to detect firewall policy violations when the network status is updated and conducts automatic and real-time violation resolutions with the help of several strategies designed for diverse network update situations.

A near real-time security system for SDN environments containing IoT devices is presented in [39]. This systems is able to detect threats by analyzing traffic and using a Convolutional Neural Network.

In summary, the analysis of the literature has shown that computer security has been extensively studied under different angles. Particular attention is dedicated nowadays to the challenges of new and emerging technologies, such as machine learning, lightweight cryptography, cloud computing and SDN, to name a few.

4.2 Machine Learning based Security

Machine learning is becoming the de-facto standard solution for many problems in the computer security field. Although its algorithms can be very effective for detecting patterns and reveal previously unseen attacks, there are many open issues to be tackled. In the literature, machine learning has been applied to a wide variety of security problems, such as spam/phishing detection, malware analysis and intrusion detection/prevention systems.

In what follows, we briefly review the state of the art in these areas. Spam can be defined as unsolicited bulk messages, typically emails, that not only waste users' time, but also consume a considerable amount of network bandwidth. In addition, some spam emails may also include malware as attachments. For these reasons, being able to discriminate between legit emails and spam is of paramount importance. Spam filtering has been studied in the literature for several decades as shown by a recent survey [40]. Unfortunately, spam is constantly evolving and increasingly taking advantage of alternative media. For example, mobile phones are used to send spam SMS messages. These messages are particularly attractive for spammers, because they are more frequently read with respect to emails. Many papers aimed at contrasting the SMS spam threat have been published. For example, Bosaeed et al. [41] proposed an efficient machine learning based classifier for SMS spam detection that can be used in cloud, fog or edge environments. In addition, they provide recommendations on the usage of spam filters and corresponding mail classifiers based on user requirements such as classification accuracy, true negatives, and computational resource requirements. SMS is not the only medium used by spammers. In fact, in the effort to evade text-based filters, spammers sometimes embed spam text in images. This kind of spam is often referred to as image spam. Sharmin et al. [42] apply convolutional neural networks for detecting image spam. The paper considers the Canny edge detector as a tool for extracting information related to edges in images. The convolutional neural networks used considered raw images, Canny images, and a combined feature consisting of both the raw and Canny images.

Another problem addressed in the literature refers to malware analysis. This problem is particularly challenging because nowadays malware can be a complex and advanced software, sometimes also engineered to attack a specific target. In addition, this type of malware can be highly persistent and able to escape different security controls. A detailed survey on sophisticated attacks and evasion techniques used by contemporary malware is presented in [43]. This study shows that there is a huge need for efficient security systems able to detect complex malware in an efficient way.

Although classification of malware is a difficult task, many works in this area have been proposed. Some works focus on specific types of malware families (see [44–46]), while others, as highlighted by a recent survey [47], specialize on the detection of the family a malware belongs to.

Another work in the area of malware detection is proposed by Alazab et al. [48] who propose a classification model for mobile devices malware that combines features related to permission requests and API calls. These features highlight a big contrast between malware and benign applications. In particular, malware often request dangerous permissions to access sensitive data. Interestingly, several of the mentioned research works on malware analysis focus on mobile malware. This suggests the significant popularity of mobile malware among malware developers.

Another approach that is gaining significant attention consists in applying machine learning for classifying malware represented as an image. This approach enables an effective application of convolutional neural networks, because of the huge number of features to evaluate. A literature analysis demonstrates that convolutional neural networks are currently widely adopted as a malware classification technique (see [49–51]). In addition, experiments on custom neural network architectures based on convolutional neural networks are also being conducted. Vasan et al. [52] proposed a multi-class classifier able to detect variants of malware families and improve malware detection using a custom CNN-based deep learning architecture. The proposed method converts the raw malware binaries into color images used by a fine-tuned CNN architecture to detect and identify malware families.

Similarly to malware detection, machine learning has also been applied in the framework of malicious traffic detection. A large variety of solutions have been developed in the form of intrusion detection and prevention systems as highlighted in recent literature surveys [53, 54].

Machine learning techniques can also be applied for building data-driven solutions that ideally adapt, to a certain degree, to previously unseen attacks. Unfortunately, there are limitations on the kind of attacks that can be detected. These limitations generally depend on the data provided to the learning algorithm. For this reason, most research papers specifically tailor intrusion detection systems to specific environments and applications, hence limiting the number and type of previously known attacks that can be detected.

Kasongo et al. [55] proposed a wireless intrusion detection system based on feed forward deep neural networks coupled with a filter-based feature selection algorithm and show that this approach outperforms other techniques.

Almiani et al. [56] focused on an intrusion system for IoT networks based on deep neural networks. The proposed model is particularly good at detecting DoS attacks.

In summary, the analysis of the state of the art has shown that machine learning is nowadays very popular for computer security. This thesis work takes also advantage of machine learning techniques to extract useful security insights from enterprise networks.

4.3 Graph Based Computer Security

Modeling attack scenarios in penetration testing activities has been originally addressed with the purpose of programmatically controlling penetration testing tools. A number of studies have focused on modeling computer security from a statistical perspective using graph theory. In fact, these models naturally fit into graphs. Early works considering attack graphs date back to the 90s and the beginning of the year 2000 [57–59]. In particular, Phillips and Swiler [57] proposed a graph-based system to study the problem of network vulnerability analysis. The proposed system could be used to test the effectiveness of security measures put in place and the possible vulnerabilities created with the changes of a setup. In particular, the graph-based system works similarly to an intrusion detection system and its knowledge is based on a database of common attacks. Network configuration and topology information are fed to the system as inputs together with information about possible attacker profiles. All this information is processed to create an attack graph describing the network and related vulnerabilities. In this context, nodes identify the stage of a possible attack, such as class of machines an attacker has accessed and its privilege level. Edges represent attacks or atomic attack steps in case of more complex attacks. Each edge is assigned a success probability or a cost that represents the level of effort for an attacker to successfully take advantage of that edge to perform an attack. Starting from this graph, shortest paths, that is, paths with the highest probability of success, are identified. This process can be useful both on the offensive side and on the defensive side. Indeed, the shortest path on an attack graph may help a penetration tester identifying the actions to perform to compromise a network. The shortest path can also highlight critical issues to be mitigated.

In the context of attack graphs, some papers focused on methodologies to automate the analysis and the creation of attack graphs. For example, Sheyner et al. [59] use symbolic model checking algorithms to automatically build and analyze attack graphs. This approach can be applied to model basic network appliances such as firewalls and intrusion detection systems. With respect to [57], this work tends to be more general and is not focused on an attack-centric view of the graph. In fact, thanks to the flexibility provided by a general symbolic modeling language, it is possible to take into account seemingly benign events, e.g., the failure of a link. Model checking is also used in [58] where vulnerabilities are encoded in a state machine description suitable for a model checker and different assertions are performed based on the attacker privilege and compromised hosts. The model checker can

either assure the security of a computer network or confute its security by providing an example where each step of a successful attack are detailed.

The approaches proposed by early works, based on a complete enumeration of the attack states are characterized by an exponential complexity. More precisely, the number of attacks highlighted through an attack graph grows exponentially with the number of available actions an attacker can perform and of machines on the network. To tackle this complexity, Jha et al. [60] present a minimization technique for decreasing the set of security measures that ensures system security. The paper provides a formal characterization of attack graphs and presents a reliability analysis together with an approach for performing a simple cost-benefit trade-off as a function of the attack likelihoods. The attack graphs are modeled as Markov Decision Processes and the intruder success probabilities for each attack represented in the graph are computed through the use of the value iteration algorithm.

A more challenging task is to incorporate into attack graphs more complex vulnerabilities such as chained weak misconfiguration (e.g., inadvertently caused by system administrators). An example of this class of vulnerabilities is the identity snowball attacks [61]. These attacks take advantage of users logged on a compromised host and launch additional attacks using the privileges of compromised users on other computers within the same network. To address the identity snowball attacks in large enterprise environments Microsoft developed Heat-Ray, a system able to reduce the number of machines that can be used to launch a large scale snowball attack. Although the results of Heat-Ray are promising, cybersecurity is well represented as a mice and cat game. Hence, defense techniques must be updated and able to promptly counteract novel attack techniques.

A key problem addressed in the literature in the framework of graph-based security is the definition of useful and effective metrics able to capture insights on a system vulnerability and quantify the risk associated with a system or an entire infrastructure. In this context, the work of Idika and Bhargava [62] presents some possible metrics that can be used on attack graphs, such as normalized mean or standard deviation of path lengths. In addition, the paper shows how these metrics can be aggregated to produce additional security indicators, since simple metrics such as shortest paths or the number of paths between an attacker and a target might be insufficient.

Obes et al. [63] proposed a complete representation of an attack model using the Planning Domain Definition Language. This representation allows the automatic generation and validation of attack paths in penetration testing scenarios. In addition, the paper presents an algorithm that transforms the results provided by common penetration testing tools (e.g., network mappers, vulnerability scanners) in the planning domain, and shows the scalability of the algorithm in medium-sized networks. The information used in this work includes target network, target host, set of ports used, fingerprinted applications and details about the operating system and kernel version.

Attack paths are also addressed by Chokshi et al. [64] where a heuristic-based attack graph generation algorithm that integrates different phases of network security assessment methodologies is proposed. The research paper

proposes the generation of an attack graph starting from a database of public exploits and scan results, provided by vulnerability assessment software, and provides recommendations on mitigations to be put in place. These mitigations are proposed after the evaluation of an optimization problem taking into account security and corresponding costs.

A more recent work [65] focuses on the assessment of cloud applications by presenting a methodology to automatically configure a testing environment and obtain a preliminary evaluation of the security level provided by the cloud application. This process takes into account the architecture of the cloud application under test and the security issues it may be subject to, including threats, attacks, vulnerabilities and weaknesses. One of the main contributions of this work is the methodology for continuous security assessment to be included in the software development life-cycle or in general in DevOps methodologies.

In the context of cloud security automation, Wang et al. [66] developed a framework for internal penetration testing of cloud infrastructures based on the big data Hadoop system. The paper investigates potential attacks starting from a single compromised internal node against the cloud system availability and performance. The paper also discusses a possible mitigation scheme for big data systems.

In the context of IoT technologies, George et al. [67] proposed a graph-based security framework for securing industrial IoT networks from vulnerability exploitations. The paper focuses on common vulnerabilities characterizing industrial IoT devices and develops a graph model where nodes represent device vulnerabilities, while the edges represent vulnerability dependencies. The proposed framework also provides an optimization platform to find the optimal trade-off between security and performance of the network. Another work in the context of IoT networks [68] focuses on the development of a framework for IoT penetration testing that automatically performs information gathering and exploitation on the target IoT devices taking advantage of wireless communication, either WiFi or Bluetooth. The system incorporates basic security guidelines according to the OWASP's Top 10 IoT Vulnerabilities[1] for implementing mitigations.

It is worth noticing that many research results regarding the automation of penetration testing techniques with graph models are published in the form of patents [69–73]. In addition, there is a significant gap between academic and industrial approaches. An example is represented by the Bloodhound project[2] that represents the state of the art in Active Directory threat modeling and penetration testing. Bloodhound takes advantage of graph theory and shortest path algorithms to identify attack paths in Active Directory environments. Bloodhound represents one of the most popular tools used nowadays by red teams to conduct vulnerability assessments on enterprise infrastructures relying on Microsoft technologies. Unfortunately, there is a lack of tools that can adapt to general scenarios, such as a general purpose computer networks or IoT infrastructures. For this reason, this thesis

[1] https://owasp.org/www-project-internet-of-things/
[2] https://bloodhound.readthedocs.io/en/latest/index.html

work focuses on providing a methodology and a framework based on graph theory to model computer networks and their properties from a security perspective. The graph modeling has been used with the purpose of providing interesting insights about critical nodes and paths in a network.

4.4 Security Tools and Frameworks

Many tools and frameworks addressing security issues in different scenarios have been proposed in the literature. For example, Ismagilova et al. [74] focused on the security of smart-city applications and presented a comprehensive information framework containing a set of guidelines for developing secure applications in the context of smart-cities. A security framework for vehicular networks that is presented in [75] is able to protect smart vehicles from external attackers by implementing a system relying on a hierarchical cooperative game that is managed by a leader software agent controlling an intrusion detection system, prevention system and reaction system.

Xi et al. [76] proposed a penetration testing framework to assess the security of power web systems that integrates property information and expert experience, thus leading to automatic vulnerability verification and exploitation. A framework that implements a security scheme with flow monitoring algorithms for fast anomaly detection and prediction of DDoS attacks on SDN is presented in [77].

Other research tracks focus on the development of frameworks for mitigating specific attacks. Giechaskiel et al. [78] proposed a framework for analyzing threat models in signal injection scenarios and introduced an algorithm for calculating the security level of real systems composed of sensors. Bhayo et al. [79] proposed a DDoS detection framework leveraging software defined IoT technologies, that is, entire IoT infrastructures solely defined and configured by software. Security aspects related to specific emerging technologies such as 5G are addressed in [80] where a framework for the formal verification of control-plane protocols spanning multiple layers of the 5G stack is presented.

In summary, there is a constant need of automating procedures in many computer security applications and scenarios. For this reason one of the outcomes of this thesis work is a framework that semi-automatically analyzes complex enterprise network security scenarios.

Chapter 5

Theoretical Background

In this chapter, the background that this thesis relies on is presented. In the first part, the main concepts of machine learning are detailed, following with a discussion of common classification algorithms with a particular focus on decision trees and random forests. These algorithms have been tested for the development of the toolset. Finally, basic notions about graph theory are provided. These notions are useful to understand graphs and shortest path algorithms, that are fundamental building blocks of this thesis work.

5.1 Introduction to Machine Learning

Machine Learning is currently a scientific field widely used for very different applications. The main objective of this field is the construction of "learners" which are models that can be used to solve problems which would be too complicated to tackle by defining a set of heuristics. A core objective of a learner is to generalize from its experience. Generalization in this context refers to the ability of a learning machine to be accurate on new, unseen examples or tasks after having experienced a training data set. The training examples come from some unknown probability distributions and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases. Recently, machine learning has gained significant popularity, due to the fact that deep learning is now computationally feasible. In this chapter, after some basic definitions, a taxonomy of machine learning algorithms is presented. Supervised learning and classification problems are also discussed.

5.1.1 Taxonomy of Machine Learning Algorithms

Machine Learning is considered a subfield of Artificial Intelligence [81]. Arthur Samuel in 1959, defined Machine Learning as the field of study that gives computers the ability to learn without being explicitly programmed [82]. While this definition is not accurate, it gives an intuition on the importance of machine learning in creating autonomous software agents. Tom Mitchell in 1998 provided a more precise definition of machine learning by first defining what a well-posed learning problem is. Mitchell stated that a computer program is said to learn from experience E with respect to some task T with a performance measure P, if its performance on T, as measured by P, improves

with experience [83]. For example, if we consider the game of chess and a software agent trying to learn how to play chess, in this case we can define:

- E: Experience of having the program play many games;
- T: Task of playing chess;
- P: Probability of winning the next game against some new opponent.

Machine learning explores the study and construction of algorithms that can learn from data and make predictions. These algorithms do not follow strictly static program instructions, instead they make data-driven predictions or decisions. Machine Learning is applied to various types of problems, such as:

- Problems whose solution requires a lot of hand tuning or long lists of rules or heuristics;
- Problems without an existing solution;
- Problems aimed at extracting useful information from large data sets;
- Problems addressing a complex and dynamic environment.

As already stated, a core objective of machine learning system is to generalize from its experience. Generalization in this context is the ability of a learning machine to be able to handle new, unseen examples or tasks after having experienced a training data set [84]. Defining as D a single observation – which is a random variable – and as N the number of total observations, generally the study of machine learning and its algorithms can be divided into three broad categories [85]:

- **supervised learning**: where a computer has to be taught a certain task. In this framework, the learning phase is based on complete observations. Each observation $\{D_1, D_2, ..., D_N\}$ includes values for all the random variables in the model, the objective is to learn a distribution Y;
- **unsupervised learning**: where computer learns by itself a certain task. In this framework, the learning phase is based on incomplete observations $\{D_1, D_2, ..., D_N\}$ and do not necessarily include values for all the random variables in the model. The objective is to learn a distribution Y;
- **reinforcement learning**: where a computer learns by itself an optimal policy. In this framework, the observations $\{D_1, D_2, ..., D_N\}$ are states or situations and at each state X_i a software agent must perform an action a_i that produces a result r_i. The objective is to define a function $a_i = \pi(D_i)$ – called policy – that describes a strategy that a software agent will follow. The strategy should be optimal, in the sense that it should maximize the expected value of a function $v(< r_1, r_2, ..., r_n >)$ which keeps track of the sequence of results.

Scientists in the area of machine learning are not very strict on the subdivision of these areas, indeed on some machine learning books [86] we may find four or five sub-categories, but generally they are slight variations of the basic areas described (e.g., sometimes we can read about semi-supervised learning [87], or recommender systems [88] as a fourth category). Also it is important to notice that this is a classification based only on a small set of criteria, (i.e., whether data is labeled and purpose of the problem), although these are the most common criteria taken into account when it comes to machine learning algorithms taxonomy, there are even other categorizations. Other examples of taxonomy attempts consider the division between on-line and off-line techniques or we instance-based and model-based learning techniques and so on and so forth.

5.1.2 Supervised Learning

Supervised learning is the machine learning task of inferring a function from labeled training data [89]. The training data consists of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario allows a supervised learning algorithm to correctly determine the result for unseen data instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. The two main problems that supervised learning algorithms solve are [90]:

- regression: a regression problem is applied when the output variable is represented by real values;
- classification: a classification problem is applied when the output variable is represented by categorical values; e.g., "red" or "blue", "disease" or "no disease", in the case of this thesis work for example, we want to be able to classify a network as either "safe" or "vulnerable".

In turn, classification algorithms can be subdivided into two main categories:

- generative algorithms: these learn a model of the joint probability distribution $p(X, y)$, where X represents the inputs and y corresponds to the output. The prediction is computed by using Bayesian rules. This requires the conditional probability distribution $p(y|X)$, and the choice of the label y. Examples of generative algorithms are: Naive Bayes, Latent Dirichlet Allocation, Probabilistic context-free grammar, Hidden Markov Models, Gaussian mixture models;
- discriminative algorithms: these classifiers model the posterior distribution [91] $p(y|X)$ directly, or learn a direct map from inputs X to

the class labels y. Examples of generative algorithms are: Logistic regression, Support Vector Machines, Maximum Entropy Markov Model, Conditional Random Fields, Neural Networks, Decision Trees.

Although there is a common belief that discriminative algorithms always work better with respect to generative algorithms, an important result of the work by Andrew Ng [92], shows that there are two distinct regimes of performance which depend on the size of the training data set. As the number of training examples increases, in the first regime of performance the generative model has already approached its asymptotic error and is performing better, while in the second regime the discriminative model approaches its lower asymptotic error and performs better. Generative models generally have an higher asymptotic error with respect to discriminative models, but they approach this limit earlier.

5.1.3 Unsupervised Learning

Unsupervised machine learning is a machine learning approach aimed at inferring a function to describe an hidden structure starting from "unlabeled" data. Hence, in this case, the supervisory signal is not included in the observations. The goal of discovering interesting structures in the data is sometimes called knowledge discovery [90]. Since the dataset processed by the unsupervised algorithms is unlabeled, it is not always possible to evaluate the accuracy of the structure that is provided as output by the algorithm. The main problems that unsupervised learning algorithms solve are:

- cluster analysis, or simply clustering, that is the process of partitioning a set of observations (i.e., data objects) into subsets. Each subset is a cluster and objects in a cluster are similar to each other, and dissimilar to objects belonging to different clusters. The set of all clusters is referred to as clustering [93];

- visualization and dimensionality reduction, that is, the process of reducing the number of random variables under consideration, by obtaining a set of principal variables (i.e., components) describing the entire dataset [94];

- association rule learning, that is, the method of discovering interesting relations between variables in large datasets. The goal of association rule learning is to identify strong rules using a measure of interestingness [95]. Based on the concept of strong rules, Rakesh Agrawal, Tomasz Imieliński and Arun Swami [96] introduced association rules for discovering patterns between purchase of products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets;

- anomaly detection, or outlier detection, that is, the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset [97]. Typically in an anomaly detection

system the anomalous items translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions.

5.1.4 Classification

In supervised learning, the dataset used to build a model is called "training set". We denote with m the number of training examples and each training example is identified with the notation:

$$\left(X^{(i)}, y^{(i)}\right) = i^{th}\, training\, example$$

where $X^{(i)}$ is called feature vector, representing an array containing values for all the features of a training example, while $y^{(i)}$ is the target value related to the specific values in $X^{(i)}$. The set of features are generally chosen by the specific domain expert.

Classification algorithms used as black boxes are very simple to understand. They expect as input a "training set", and give us as output an hypothesis function or classifier. The hypothesis function, that is a function of the training examples and of the model parameters, will be the one which will represent the classification result. Generally an hypothesis function can be denoted as:

$$h_\theta = g\left(\Theta, X\right)$$

where g is a function that depends on the used classification algorithm. Classification algorithms have to find the parameters of the model, denoted in the above formula with vector Θ such that the cost function is minimized with respect to vector Θ. Classification can hence be considered an optimization problem. In fact, in order to minimize the error between the classifier $h\left(x\right)$ and the supervisory signal y, a minimization problem has to be solved. In particular, To quantify the distance between h_Θ and y, it is necessary to define and compute the error. In the context of classification, the error is called "misclassification error" and can be mathematically defined as:

$$err\left(h_\theta\left(X\right), y\right) = \begin{cases} 1 & \text{if classification error} \\ 0 & \text{otherwise} \end{cases}$$

The first goal to be achieved for a classifier is in general to minimize the misclassification error function on the training examples. In this context, we can notice that there are similarities between the problem of classification and regression. In fact, an analogy between regression and classification can be made, if we consider instead of the misclassification error function, another cost function, as for example:

$$J\left(\Theta\right) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta\left(X^{(i)}\right) - y^{(i)}\right)^2$$

Note that the above cost function is commonly used in regression problems but the measure of distance in this case is continuous. The purpose of the $\frac{1}{m}$ term in regression is to remove the dependency on the data set size. The error magnitude is thereby independent from m. The accuracy of the error estimate increases as m increases, assuming that data examples are selected randomly.

5.1.5 Classification Performance

To evaluate classification performances, it is possible to compute two different metrics, namely:

- Accuracy, appropriate for balanced datasets;
- Confusion Matrix, F_1-Score, precision and recall, appropriate for unbalanced datasets.

A dataset is balanced when the number of target values for all categories are equal, while a dataset is unbalanced when the number of target values belonging to different categories are different. If the dataset is balanced we can use the accuracy as measure for classification. These metrics should be computed on a dataset which is not the one used for training. Usually the original dataset is divided into two or more subsets. Common splits are, for example, 70% of dataset used for training, and 30% used for validation, or 80% used for training and 20% used for validation. Let us remark that for practical reasons it is a good idea to randomly shuffle the data before splitting. After data has been split, we cannot test our model on the training data, since in this way we cannot be sure about how the model generalizes on new examples. Hence, after the model has been trained we can measure the performance of it by measuring the accuracy on the test dataset. Accuracy is defined as:

$$A = \frac{N_c}{N}$$

where N_c is the number of correctly classified observations of the validation set, and N is the total number of observations in the validation dataset.

It is reasonable to think that this measure does not make sense for unbalanced dataset.

For unbalanced dataset the evaluation relies on different metrics, such as precision, recall, F_1-score and confusion matrix [98]. In details, the metrics are defined as follows:

- Precision (also called positive predictive value) is defined as the ratio of negative-labeled data being correctly classified. It is the fraction of relevant observations among the retrieved observations, that is:

$$P = \frac{true\ positives}{predicted\ positives} = \frac{true\ positives}{true\ positives + false\ positives}$$

- Recall (also known as sensitivity) is defined as the ratio of positive-labeled observations being correctly classified. It is the fraction of relevant observations that have been retrieved over the total amount of observations, that is:

$$R = \frac{true\ positives}{actual\ positives} = \frac{true\ positives}{true\ positives + false\ negatives}$$

A classifier with high precision and high recall is in general a good classifier.

Let us remark that precision and recall are generally better indicators for the representation of the performance of a classifier when the dataset is unbalanced. Moreover, another common performance indicator whose attempt is to combine precision and recall in single metric when dealing with unbalanced datasets, is the so called F_1 score. The F_1 score is defined as:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

In addition to the above defined metrics, a less commonly used performance indicator is the confusion matrix. A confusion matrix, also known as error matrix, is a specific table that allows visualization of the performance of a supervised learning algorithm. Each row of the confusion matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa) [99]. The confusion matrix is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table). In this contingency table all correct predictions are located in the diagonal of the matrix, so it is easy to perform a visual inspection for prediction errors.

5.1.6 Tuning Machine Learning Systems

There is an important problem to be addressed in many machine learning problems, that is the tuning of the model. Tuning a model essentially means selecting the best parameters for an algorithm in order to optimize its performance. The following are possible guidelines to take into account in order to reduce classification errors and improve the performance:

- increase the number of training examples;
- decrease the size of the set of features;
- increase the size of the set of features;
- add polynomial features (x_1^2, x_2^2,...);
- control the overfitting/underfitting of the model.

In addition machine learning diagnostics can be applied to gain insights about what is or is not working within a learning algorithm and gain guidance as to how best improve performance. It becomes important at this point to define the concepts of bias and variance.

5.1.7 The Bias vs Variance Tradeoff

In statistics and machine learning, the bias–variance trade off (or dilemma) is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set [100]. More precisely:

- The bias is an error originated from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting);
- The variance is an error originated from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

We define $J_{train}(\Theta)$ and $J_{CV}(\Theta)$ respectively as the misclassification error on the training set and on the validation set.
The symptoms in case of underfitting are:

- the error $J_{train}(\Theta)$ on the training set is high;
- the error $J_{CV}(\Theta)$ on the validation set is high and is approximately similar to $J_{train}(\Theta)$.

The symptoms in case of overfitting are:

- the error $J_{train}(\Theta)$ on the training set is low;
- the error $J_{CV}(\Theta)$ on the validation set is significantly higher than the error on the training set $J_{train}(\Theta)$.

The analysis of bias and variance tells us whether collecting more data makes sense. Indeed, huge datasets are not always helpful for classifiers. In particular, if a learning algorithm has a high bias, and hence $J_{CV}(\theta)$ is almost equal to $J_{train}(\theta)$ for $m \to \infty$, a larger training dataset will not help much by itself. In addition, if a learning algorithm is suffering from high variance, and hence $J_{CV}(\theta)$ is much larger than $J_{train}(\theta)$ for $m \to \infty$, a larger training dataset is likely to help.

5.1.8 Hypothesis Evaluation

In all those cases when data cannot be plotted, for example when a dataset has many features, it is still possible to understand if we are overfitting or underfitting without relying on the above mentioned plots. What is generally done in these cases is to first split the dataset into "training set" and "test set"; Then what is done is:

- learning the parameter Θ from the training data (minimizing training error $J(\Theta)$) (e.g., on the 70% split);
- compute test set error, in case of classifiers this is just the misclassification error, defined previously, which is given by:

$$error_{test} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err\left(h_\theta\left(X_{test}^{(i)}\right), y^{(i)}\right)$$

Once, the error has been computed, we can use "learning curves" to understand if we are underfitting or overfitting.

5.1.9 Learning Curves

A fundamental tool used in the visualization and troubleshooting of machine learning systems is the so called "learning curve". A learning curve is a graphical representation of the increase (or decrease) of learning (vertical axis) as a function of the experience (horizontal axis). Ideally we should have learning curves with a low J_{train} and J_{cv} and with J_{cv} approaching J_{train}. Examples of learning curves can be observed in Figures 5.1 and 5.2. We can see in those figures two examples of non desirable conditions. In particular, Figure 5.1 shows situation characterized by a high variance, hence in this case collecting more data would help to increase the performance of the corresponding system. On the other hand, Figure 5.2 shows a situation characterized by a high bias situation, so more data in this specific case will not actually help to increase the performance of the model under exam.

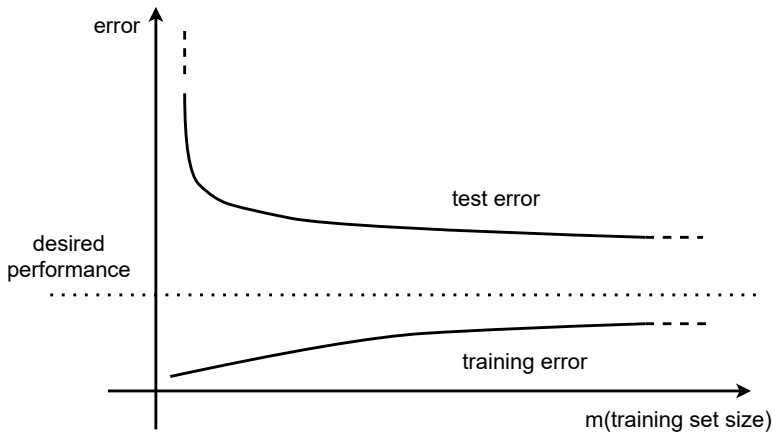


FIGURE 5.1: Typical learning curve for high variance

5.1.10 Model Selection and Additional Datasets

The "model selection" phase is related to the choice of the features and of the model parameters used to train the model. Model selection helps us in the choice of the optimal subset of features, or on the degree of the polynomial used by the model to fit the data. Basically, we run into model selection

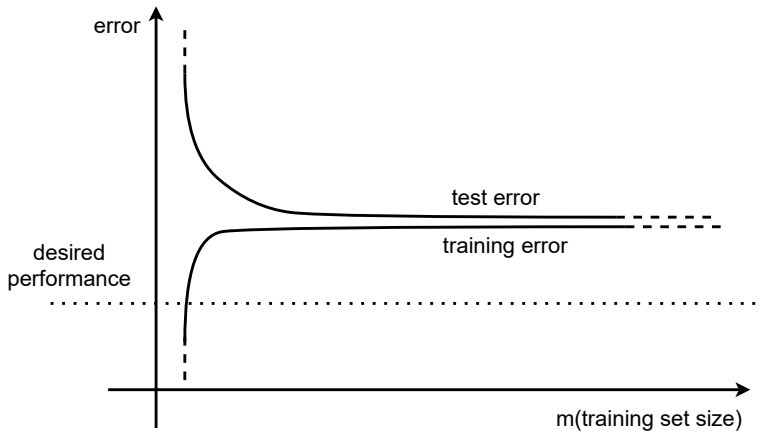


FIGURE 5.2: Typical learning curve for high bias

methodologies whenever we want to understand what are the best parameters for our model.

A good approach for selecting a model is through the use of the so called "cross-validation" datasets. Basically, this consists in dividing our dataset into three or more partitions. The number of partitions depend on the number of parameters we want to tune, but basically to tune a single parameter we would split a dataset as follows:

- training set: 60%;
- cross validation set (or simply validation set or cv-set): 20%;
- test set: 20%.

To make an example, let's say we want to understand which is the best polynomial to find Θ starting from our training set, by using different polynomials. Basically we want to tune our model to find the best polynomial. Using different tuning values we would find different θ values for different polynomial degrees:

- $h_\theta(x) = \theta_0 + \theta_1 \cdot x$
- $h_\theta(x) = \theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2$
- ...
- $h_\theta(x) = \theta_0 + \theta_1 \cdot x + ... + \theta_{10} \cdot x^{10}$

In this context, since we want to test the performance of our model with different parameters it is not enough to compute the errors only by relying on the test dataset (using e.g., a split 80/20). In fact, when selecting a model, it is always necessary to use at least an additional dataset known as "cross-validation dataset". This dataset is used to find the best set of parameters for our model, and once these are found we can finally test its performances on the test dataset. In fact, using solely a test dataset when tuning a model,

would not give a precise indication of how well our model generalizes on unseen data, and in these cases the performance of the system may be overestimated. This is a common mistake when using machine learning algorithms.

In conclusion, let us remark that whenever we are tuning a model, it is considered best practice to split the dataset into at least three partitions, and proceed as follows:

- train the model on the training set;
- tune the model with respect to a parameter using a cross-validation set;
- check the performance of the model the test set.

Note that, the more parameters we want to tune, the more cross-validation datasets we have to use.

5.1.11 Logistic Regression

Logistic regression is an old standard statistical classification method, that is commonly used for many classification problems and particularly appropriate for models involving binary decisions, although it can be generalized to multi-class problems [101]. This classification method relies on the logistic model (or logit model) which is used to model the probability of a certain class or event existing such as pass/fail, safe/dangerous or healthy/sick. Each possible output class would be assigned a probability between 0 and 1, with a sum of one.

Logistic regression is very similar to a certain extent to linear regression, the fundamental difference consists in the type of output that we are trying to predict and on the cost function.

In fact, the classification problem could be approached by ignoring the fact that y is discrete-valued, and use the linear regression algorithm to predict y given x. However, this method would perform very poorly in this context. In addition, it also wouldn't make sense for an hypothesis function $h_\theta(x)$ to take values larger than 1 or smaller than 0, as it may happen with linear regression, when we know that $y \in 0, 1$. In order to tackle a discrete-valued output, logistic regression involves a particular choice for the hypotheses function $h_\theta(x)$. More precisely:

$$h_\theta(x) = g\left(\theta^T \cdot x\right) = \frac{1}{1 + e^{-\theta^T x}}$$

where:

$$g_z = \frac{1}{1 + e^{-z}}$$

is called the logistic function, or sigmoid function. The shape of this function is depicted in Figure 5.3. As can be seen Notice that when $z \to \infty$, $g(z)$ tends towards 1, while $g(z)$ tends towards 0 as $z \to -\infty$. Moreover, $g(z)$, and as consequence also $h(x)$, is always bounded between 0 and 1. We keep the convention of letting $x_0 = 1$, so that: $\theta^T \cdot x = \theta_0 + \sum_{j=1}^{n} \theta_j \cdot x_j$. Although other functions that smoothly increase from 0 to 1 could be used, the sigmoid

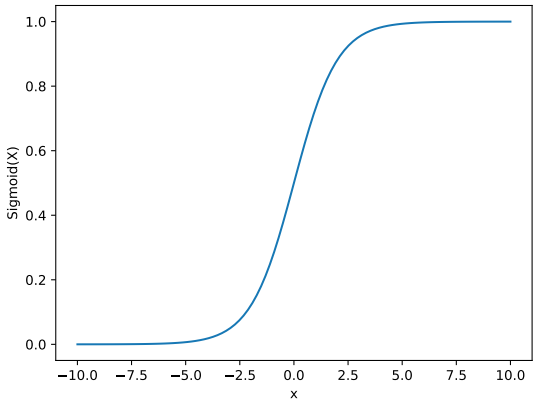


FIGURE 5.3: Sigmoid function

function other than being a natural choice also has several useful properties. In fact, the derivative of $g(z)$ can be computed as:

$$\begin{aligned} g'(z') &= \frac{d}{dz} \cdot \frac{1}{1+e^{-z}} \\ &= \frac{d}{dz} \cdot \frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2} \cdot (e^{-z}) \\ &= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z) \cdot (1 - g(z)) \end{aligned}$$

This derivative as we will allows the computation of the stochastic gradient descent, that is a common optimization algorithm used in different machine learning applications. Indeed, in order to fit for θ, a least squares regression can be derived by employing the maximum likelihood estimator under a set of assumptions. In what follows, the classification model is endowed with a set of probabilistic assumptions and the parameters are fit by using the maximum likelihood technique.

Let us assume that:

$$P(y = 1|x;\theta) = h_\theta(x) \, P(y = 0|x;\theta) = 1 - h_\theta(x)$$

Let us remark that this can also be written more compactly as:

$$P(y|x;\theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

With the assumption that the m training examples of data are independent, we can derive the likelihood of the parameters as:

$$L(\theta) = \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)};\theta\right)$$

At this point it is easier to maximize the log likelihood, hence:

$$l(\theta) = \log L(\theta)$$

This derivation is very similar to the case of linear regression, hence the gradient descent technique can be used. Each update can therefore be described in the vectorial notation as $\theta := \theta + \alpha\delta_\theta l(\theta)$.

Finally, the stochastic gradient ascent rule in the case of logistic regression corresponds to:

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j^{(i)}$$

Let us remark that this derivation is almost identical to the linear regression one. The fundamental difference in this case is related to $h_\theta\left(x^{(i)}\right)$ that is defined as a non-linear function of $\theta^T x^{(i)}$. Nonetheless, gradient descent optimization technique can be applied together with the same update rules.

Another difference with respect to the linear regression is related to the interpretation of the coefficients of the mode. In fact, the coefficients in the logistic version are more difficult to interpret with respect to the ordinary linear regression. More precisely, the interpretation could correspond to the changes in the log-odds of the outcome being modeled, but the meaning of these log-odds is a little opaque since practically speaking the effect on the probability that moving one of the input features will have depends on other factors. However, the direction of the coefficients alone can be interpreted usefully. In fact, features with positive coefficients increase the probability of the modeled outcome as they increase, while features with negative coefficients decrease the probability of the outcome as they increase.

5.1.12 Support-Vector Machine

Support-vector machine (SVM) is a non-probabilistic binary linear classifier frequently used in machine learning. The non-probabilistic aspect is its major key strength. In fact, this aspect is in contrast with probabilistic classifiers such as the Naïve Bayes. More precisely, an SVM works by separating data across a decision boundary, that can be defined as "plane". A plane is determined by a subset of the data, known as "feature vectors". The data subset that supports the plane are known as "support vectors". The remaining part of the dataset does not have any influence in determining the position of the decision boundary in the feature space.

In contrast with SVMs, probabilistic classifiers develop a model that best explains the data by considering all of the data instead of just a subset. In

addition, in general SVM require less computing resources with respect to probabilistic classification techniques.

A depiction of the separation of data at the base of the SVM inner working is shown in Figure 5.4. The image shows how the hyperplane separates sets

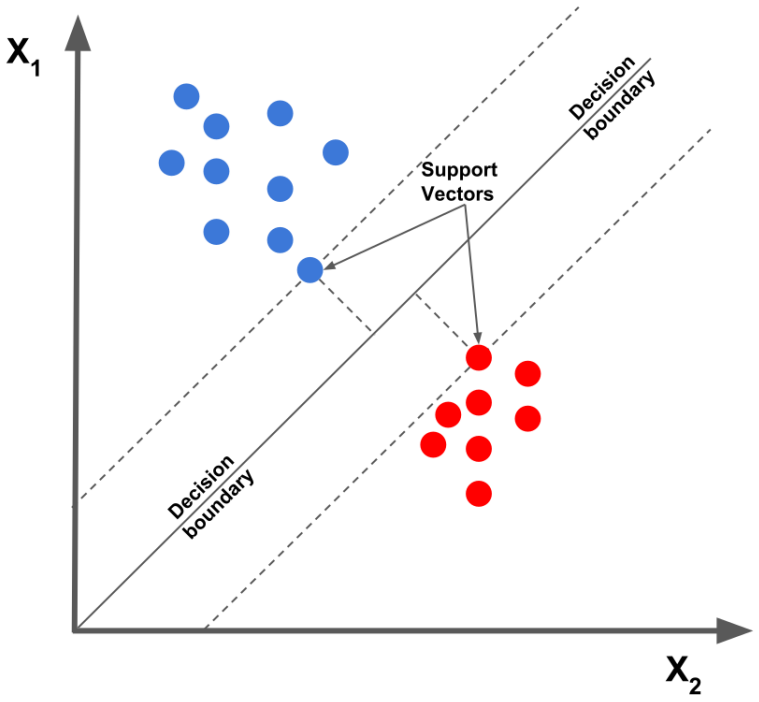


FIGURE 5.4: Support-Vector Machine hyperplane separating input data

of data belonging to different classes (i.e., characterized by different colors). In fact, the final aim of SVM is to find the optimal hyperplane that separates data belonging to different classes by maximizing the margin between the hyperplane and the data samples.

More precisely, an SVM is fed with a training dataset of n points in the form $(x_1, y_1), ..., (x_n, y_n)$, with y_i being either 1 or -1 corresponding to the class to which the point x_i belongs. More precisely, each x_i corresponds to a p-dimensional vector. The aim of SVM is to find the "maximum-margin hyperplane" that separates the group of vectors x_i for which $y_i = 1$ from the group of points for which $y_i = -1$. This maximum-margin hyperplane is defined in order to have the distance between the hyperplane and the nearest point x_i from either group maximized.

Any hyperplane can be defined as the set of points x satisfying the following equation:

$$w^T x - b = 0$$

where w is the normal vector to the hyperplane and $\frac{b}{||w||}$ determines the offset of the hyperplane from the origin along the vector w.

In general, performing a classification task with an SVM classifier corresponds to minimizing an expression of the following form:

$$\left(\frac{1}{n}\sum_{i=1}^{n} max\left(0, 1 - y_i\left(w^T x_i - b\right)\right)\right) + \lambda||w||^2$$

Where λ determines the trade-off between increasing the margin size and ensuring that the x_i lie on the correct side of the margin.

Note that the binary and linear aspects associated with SVM represent its limitations. A common technique used to overcome the linear limitation is to use the well-known "Kernel Trick" that is able to address the linearity restriction on the decision boundary [102].

On the other hand, the inability to deal with multinomial classification problems, that is, classify data into more than two classes, is still an area of ongoing research.

The current solution to this issue nowadays involves the creation of multiple binary SVM classifiers that compare data objects in a variety of ways, such as one-versus-all (OVA) or all-versus-all (AVA) [103]. For a problem involving k classes of data, OVA requires training k classifiers so that each class discriminates against the remaining $k-1$ classes. In particular, AVA requires $k\left(k-1\right)/2$ classifiers because each class is discriminated against every other class, and all possible pairings must be taken into account. Once all the binary classifiers for either method are constructed, a new object is classified based on the comparison that provides the largest discriminant output value.

Note that in addition to these issues for which workarounds have been found, there is another potential drawback to consider when using SVMs. In fact, the parameters of a solved model are of difficult interpretation. Nonetheless, SVMs represent a widely used solution for many problems and by simultaneously minimizing the empirical classification error and maximizing the geometric margin they are able to outperform other algorithms in many fields.

5.2 Decision Trees and Random Forests

In this chapter we delve into the tree based methods. These methods are largely applied in supervised learning both for regression and classification purposes and are known for their good performance in many applications. After the definition of the decision trees, ensemble learning methods are briefly introduced, subsequently random forests are described.

5.2.1 Decision Trees

A decision tree is a decision support tool that has a flowchart-like tree structure, where each internal node (i.e., non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) represents a specific class or class distributions [93]. The tree consists of one or more nodes, where each node, except the root, has an incoming edge. A node without any outgoing edge is called leaf node. The topmost node in a tree is defined as root node, that is, the only node without any incoming edge. An example of decision tree is shown in Figure 5.5.

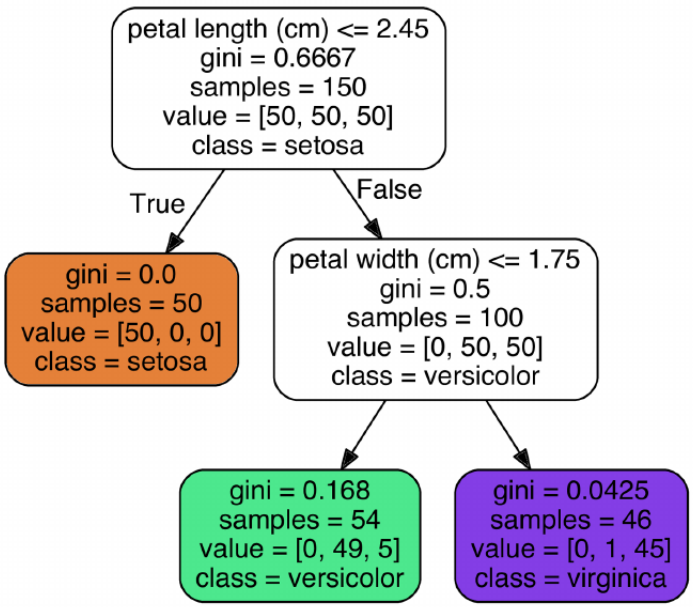


FIGURE 5.5: Example of a decision tree

In order to understand how a decision tree is used to make predictions, we consider the simple tree in Figure 5.5 who models a decision tree related to the famous Iris dataset [104]. Suppose new data of an iris flower needs to be classified. Starting from the root node (depth 0 of the tree), this node checks whether the flower's petal length is shorter than the specified length. If this is true, we move down to the left child node (depth 1). Since this is a leaf node, the predicted class is specified into the node, i.e., class=setosa.

Each node has different attributes; for example, the node samples count the number of training instances from the dataset it applies to. Another attribute is the value which indicates the number of training instances of each

class the node applies to. Finally another important attribute usually associated to a node is the Gini impurity index [105].

The Gini impurity index for a node is computed as follows:

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

where n is the number of classes, and $p_{i,k}$ is the ratio of class k instances relative to the i^{th} node. As example, the Gini impurity index of the depth-2 left node in Figure 5.5 is equal to:

$$G_i = 1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$$

A node is said to be "pure" – with a Gini impurity index equal to zero – if all dataset instances it applies to belong to the same class. Compared to other classifiers decision trees are very intuitive to understand and their decisions (e.g., classifications) are easy to interpret. These models are often referred to as "white box models". In contrast, neural networks and random forests are generally considered as "black box models". In fact, even if these are very good classifiers, it is very difficult to explain in simple terms why a specific prediction was made. Decision trees can also estimate the probabilities of an instance to belong to a particular class k. For example, the ratio between the number of instances of a specific class associated to a leaf node and the total number of training instances associated to the node gives us the probability that a specific training instance related to the node belongs to the specific class. For the decision tree shown in Figure 5.5 and a flower whose petals are 5 cm long and 1.5 cm wide, the resulting leaf node is the depth-2 left node. The decision tree provides the following probabilities: 0 for Setosa, 0.907 for Versicolor and 0.093 for Virginica. The response of course provides as output the class with the highest probability.

5.2.2 CART Training Algorithm

A very common algorithm used to train decision trees is the Classification And Regression Tree (CART) algorithm [106]. This algorithm recursively splits the training set in two subsets using a feature k and a threshold t_k. In order to determine the order of the pairs (k, t_k), the algorithm selects the pair that produces the purest subsets in terms of Gini impurity index, weighted by the size of the subset.

As many classification algorithms, for decision trees the classification problem is a cost function minimization problem. The cost function for CART is given by:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

where G_{left}/G_{right} denote the measures of impurity of the left/right subsets respectively, while m_{left}/m_{right} represent the number of instances belonging to the left/right subsets respectively. The split procedure is recursive and stops when one of following conditions is satisfied:

1. the predefined maximum depth has been reached, this is a tuning parameter for decision trees;
2. a split that reduces the impurity cannot be found;
3. Additional user-defined parameters that are used to control the stopping conditions of the algorithm.

The CART algorithm is a greedy algorithm, since it searches for an optimum split at the top level, then repeats the process at each level, without checking whether a specific split leads to the lowest impurity level. Greedy algorithms generally produce a reasonably good solution, without any guarantee of their optimality. Unfortunately, training an optimal decision tree is an NP-Complete problem that requires $O\left(exp\left(m\right)\right)$ time. Hence, the problem is computationally unfeasible even for small training sets. Although there are algorithms, such as ID3 [107] which work with non binary trees the CART algorithm only works with binary trees – so traversing a tree has a computational complexity of $O\left(log_2\left(m\right)\right)$ – and the predictions are very fast even when dealing with large training datasets.

For what concerns the training of the algorithm, that is, the growing phase, the complexity is of $O\left(n \times m \cdot log\left(m\right)\right)$, where n denotes the number of features.

5.2.3 Decision Trees Parameters

Unconstrained decision trees adapt themselves to the training data, this model is non-parametric, and a common problem associated to it is being subjected to overfitting scenarios. In fact, when the number of nodes is high related to the number of features we may have an high variance model. To avoid overfitting issues, the growth of the tree can be restricted, this is known as "regularization" process. The regularization parameters depend on the algorithm used. Setting properly a maximum depth can indeed prevent model overfitting. Moreover, there are even other parameters that can be used to tune a decision tree, such as [86]:

- minimum sample split: the minimum number of samples a node must have before it can be split;
- minimum leaf number: the minimum number of samples a leaf must own;
- maximum leaf number: the maximum number of leaf nodes;
- maximum features: the maximum number of features that are evaluated for splitting at each node.

In general to regularize a model, it is possible to decrease the value corresponding to the "maximum" parameters or increase the value corresponding to the "minimum" parameters. Many algorithms first train the decision tree

without any restrictions – as a non parametric model – and then delete unnecessary nodes. The node deletion is called "pruning", and it represents an example of regularization process for decision trees. Although decision trees represent a very flexible, easy to interpret and powerful machine learning algorithm, they have some limitations. In particular, decision trees are affected by the following weaknesses:

- orthogonal decision boundaries: every split is perpendicular to one of the axis. This makes decision trees very sensitive to dataset rotations, and prone to overfitting;
- sensitiveness to small dataset variations: even removing a single element in the training set, could lead to an extremely different tree. Random forests – which will be discussed later in this thesis work – can limit this instability by averaging predictions over many trees.

5.2.4 Ensemble methods

Ensemble methods are techniques used to create multiple models by means of learning algorithms in order to obtain better predictive performance [108]. The basic principle is that the aggregation of a group of predictors provides better performance than considering individual isolated predictors [109] [110]. A set of predictors is called ensemble. The most popular ensemble methods are, bagging/pasting, boosting, stacking and random forests.

5.2.5 Voting Ensemble Techniques

An example of ensemble technique is the Voting classifier. The concept behind these voting classifiers is simple and is based on the aggregation of the predictions of each classifier. The class with the largest number of votes or with the most significant votes is then predicted. Voting classification techniques can be divided into:

- hard voting classifiers: they consider the number of occurrences of a particular output with the maximum number of votes;
- soft voting classifiers: all classifiers in the ensemble are able to estimate class probabilities. They predict the class with the highest probability, averaged over all the individual classifiers.

Soft voting classifiers often achieve better performance than hard voting because they consider more important votes with higher probabilities. In general, hard voting is useful when all classifiers have comparable performance. On the contrary, it does not make sense in case of very unbalanced classifiers, i.e., classifiers with very different accuracies, especially if the number of classifiers with higher performance are the minority in the ensemble. An example of voting classifier is shown in Figure 5.6. In this case a support vector machine, a logistic regression, a decision tree and other classification algorithms are used, then the final result is given by the output with the highest mode.

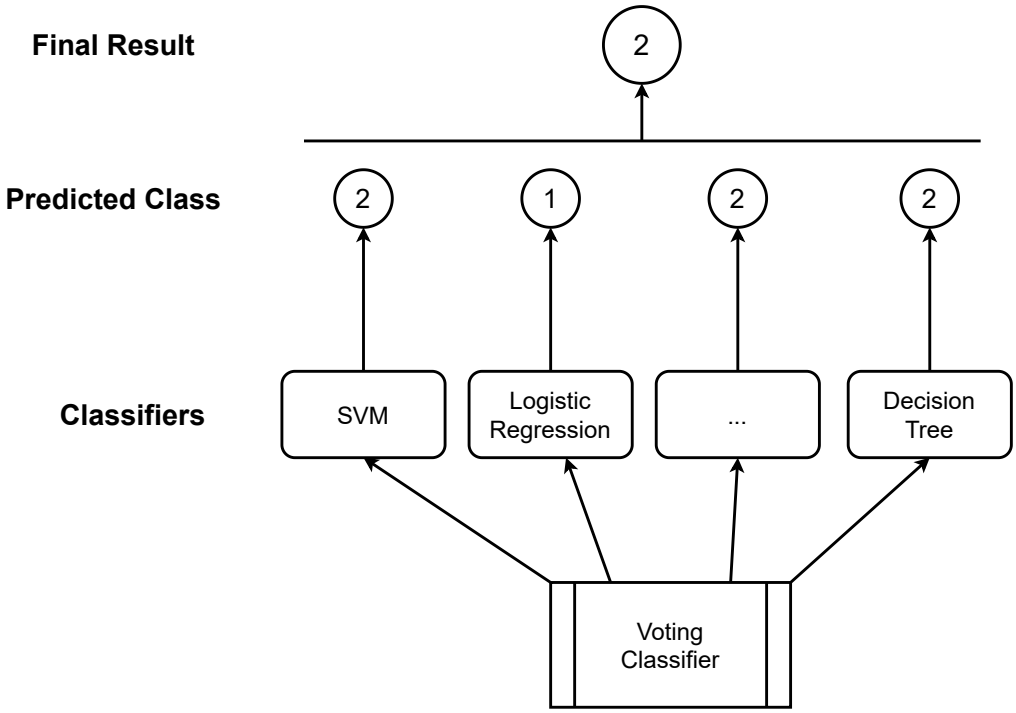


FIGURE 5.6: Example of voting ensemble classification

5.2.6 Bagging and Pasting

Bagging and pasting methods use the same algorithm in an ensemble for every predictor but on different subsets of the training dataset. In detail:

- Bagging [111]: (i.e., bootstrap aggregating), the sampling from the training dataset allows replacement;
- Pasting [112]: the sampling allows replacement.

Hence bagging allows training dataset objects to be sampled multiple times for the same classifier.

Once all classifiers are trained, all the classifiers predictions are aggregated to make a new prediction. The aggregation technique is similar to the voting classifier; in the case of a regression it is computed as the average of all regressors. When these techniques are used, it can be noticed, that each individual classifier has higher bias than if it was trained on the entire training set. Anyway, aggregation is able to reduce both bias and variance. In general the entire ensemble classifier has a lower variance but a similar bias with respect to each single individual classifier trained on the entire dataset.

By default a bagging classifier gives to each predictor only a portion of the entire training set, while the remaining part that is called out-of-bag (OOB) set is not provided to the classifier. Since a predictor never samples the OOB instances during the training phase, its performance can be evaluated on the OOB set, without the need of an additional validation set.

The performance of a bagging method is evaluated by taking the average performance of each classifier evaluated on the OOB set.

5.2.7 Random Forests

Random forests or random decision forests [113] are an ensemble learning method mainly used for classification and regression, that operate by constructing different decision trees at training time and by giving as output the class that is the mode of the classes. In case of a regression problem, the output is represented by the mean prediction of the individual trees [114].

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, relying on a set of decision trees. Given a training set $X = x_1, x_2, ..., x_n$ with responses (i.e., target values) $Y = y_1, ..., y_n$, the bagging procedure selects B times a random sample with replacement of the training set and trains trees with these samples. For $b = 1, ..., B$ the following steps are executed:

1. sample with replacement of n training examples from X, Y, that is, X_b, Y_b;
2. train a classification (or regression) tree f_b on X_b, Y_b.

After training, predictions for validation samples x_v can be made. In the case of a classification problem the output is just the mode value of all the classification trees, that is the majority vote, while in the case of a regression problem the output corresponds to the average of all the predictions from the individual trees computed as follows

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b \left(x_v \right)$$

This bootstrapping procedure leads to improvements in the model performance because it decreases the variance of the model, without increasing the bias. Thus, while one of the decision trees weaknesses is the high sensitivity to noise in training set, the average of many trees is not, as long as the trees are not correlated. Bootstrap sampling is a technique to remove the correlation between the decision trees by training them using different training sets. When all the trees are trained on a single training set, the result is a set of strongly correlated trees.

5.3 Introduction to Graph Theory

Graph Theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects [115].

Graph theory is commonly used to solve routing, network problems and is being employed to find a "best" route. Best in this case may mean the least expensive, the one involving the least amount of time or the one with the least distance. Some examples of routing problems are routes covered by postal workers, UPS drivers, police officers, garbage disposal personnel, tour buses and other applications. Furthermore, some examples of network problems are the deployment of telephone networks, railway systems, road

planning for traffic engineering, pipelines, and design of computer chips. In this chapter, after some basic definitions, a taxonomy of shortest paths techniques is presented.

5.3.1 Basic Definitions and Concepts

A graph is a pair $G = (V, E)$ of sets that satisfy $E \subseteq [V]^2$, that is, the elements of E are subsets of V composed by 2 elements [116]. The elements of V are known as "vertices" (or nodes) of the graph G, while the elements of E are known as the graph "edges" (or lines). A common graphical way to represent a graph is by using a dot for each vertex of the graph and joining each of these vertices by a line if they form an edge. An example of graph can be seen in Figure 5.7. Note that although representing a graph seems quite intuitive and simple, the representation of dynamic graphs (i.e., graphs whose structure may change with time) is a current topic of research [117].

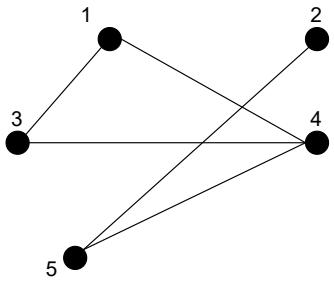


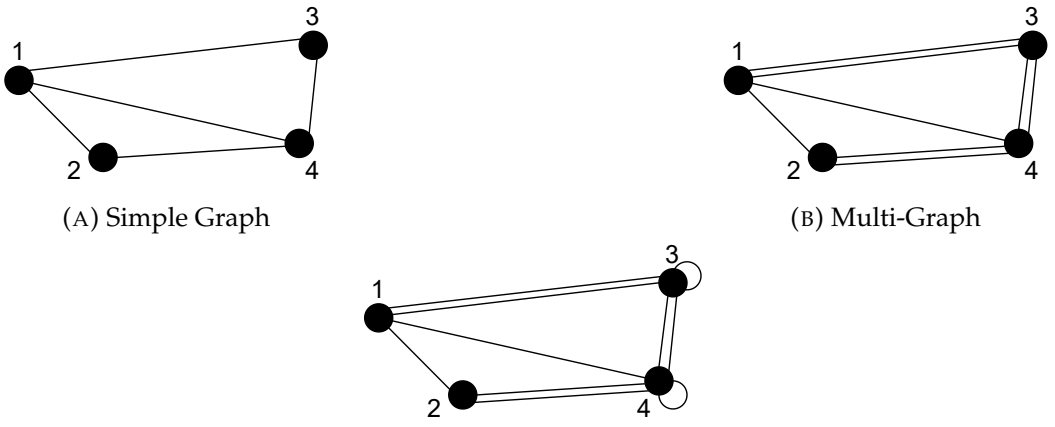
FIGURE 5.7: Basic example of graph

Graphs are commonly classified depending on the characteristics of the relationships between nodes, in particular, we can distinguish between:

- Simple Graphs, that is, graphs in which node pairs can only have a single relationship between them;
- Multi-Graphs, that is, graphs in which node pairs can have multiple relationships between them;
- Pseudo-Graphs, that is, multi-graphs where nodes can have relationships looping on themselves.

An example depicting the differences between these classes of graphs is shown in Figure 5.8.

The vertex set of a graph G can be denoted with the notation $V(G)$ (or $v \in G$), while the edge set of the same graph can be referred to as $E(G)$ (or $e \in G$). The number of vertices in a graph G is defined as its "order" and is denoted as $|G|$. A graph with an empty set of vertices and edges $G = (\emptyset, \emptyset)$ is defined as "empty graph". Given two vertices x, y of G, these vertices are "adjacent" (or neighbors) if x and y represent an edge of G. The set of neighbors of a vertex v on the graph G is denoted by by $N(v)$. We can define the "degree" of a vertex v on a graph G as $d(v) = |E(v)|$, that is, the number of neighbors of v. For a graph G we can define its minimum degree,



(A) Simple Graph

(B) Multi-Graph

(C) Pseudo-Graph

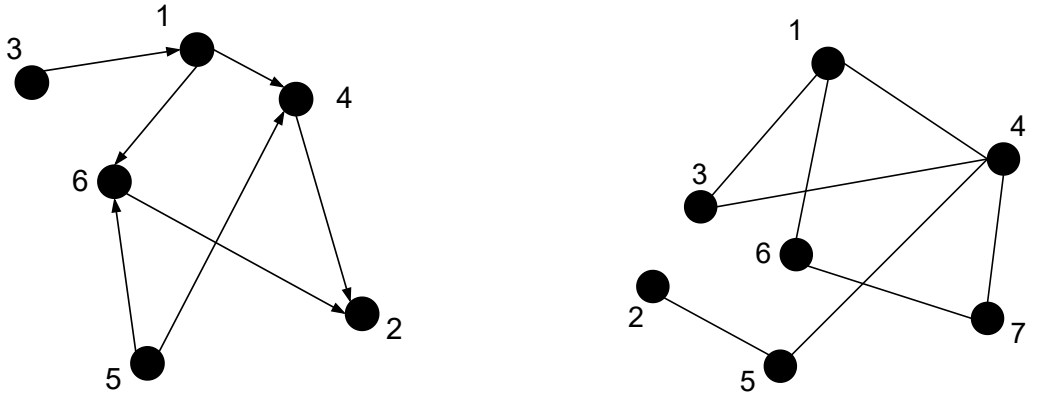
FIGURE 5.8: Comparison between different classes of graphs

that is, the node belonging to G with the lowest degree and its maximum degree, that corresponds to the node of the same graph with the maximum degree. For a graph G we can additionally define the "average degree" by computing:

$$d(G) := \frac{1}{|V|} \sum_{v \in V} d(v)$$

The average degree is a common statistic used to describe the characteristics of a graph.

Another distinction in the field of graph theory is made between "undirected graphs", where all the edges link two vertices symmetrically, and "directed graphs", where edges link two vertices asymmetrically. An example of this distinction can be observed in Figure 5.9.



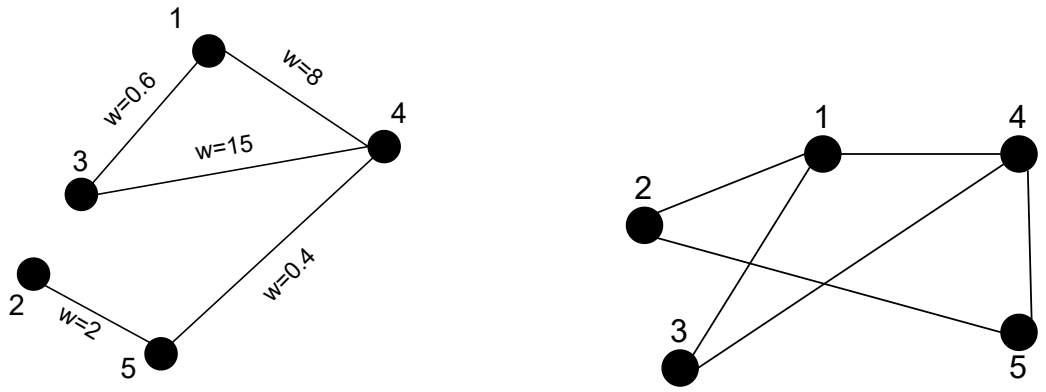
(A) Directed Graph

(B) Undirected Graph

FIGURE 5.9: Comparison between a directed and an undirected graph

Another common characterization of graphs is based on whether the edges of the graph have an associated weight or not. In the first case we can talk about "weighted" graphs, while in the other case the graph can be described

as "unweighted". An example of this distinction can be observed in Figure 5.10.



(A) Weighted Graph (B) Unweighted Graph

FIGURE 5.10: Comparison between a weighted graph and an unweighted graph

The weight associated to an edge usually represents:

- Cost or Distance, that is, the amount of effort needed to travel from one place to another;
- Capacity, that is, the maximum amount of flow that can be transported from one node (e.g., a place) to another.

On both weighted and unweighted graphs we can define one of the most important concepts in the field of graph theory, that is, the concept of path.

A path is defined as a non-empty graph $P = (V, E)$ where:

$$V = \{x_0, x_1, ..., x_n\}$$

$$E = \{x_0x_1, x_1x_2, ..., x_{n-1}x_n\}$$

with all x_i distinct, while the vertices x_0 and x_k are linked by P and are by definition known as the "ends" of the path. Given a path P we can define its length as the number of edges contained in the graph; a path of length n is denoted as P^k. An example of path is showed in Figure 5.11.

Another important concept linked to paths is definition of connectivity. A non-empty graph G is said to be "connected" if any two of its vertices are linked by a path in G. In practical terms, if there is at least one way to get from one vertex of a graph to all the other vertices of the graph, then the graph is connected. On the contrary, if there is even one vertex of a graph that cannot be reached from every other vertex, then the graph is disconnected. A comparison between an example of connected graph and a disconnected graph is depicted in Figure 5.12.

Considering the set of all paths in a graph, we can determine if a graph is cyclic or acyclic. A graph is said to be "cyclic" if it contains a cycle. A cycle is a graph built starting from a path $P = x_0...x_{k-1}$, with $k \geq 3$, more precisely

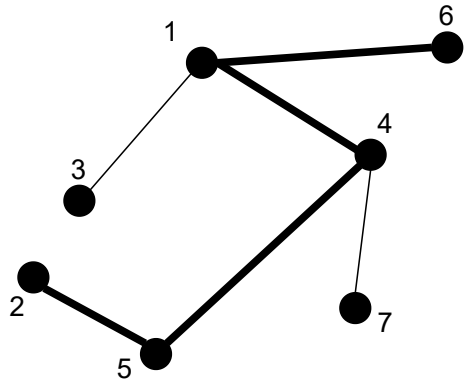
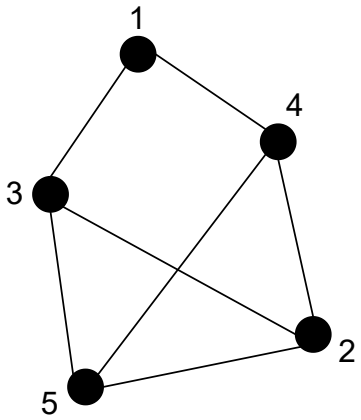
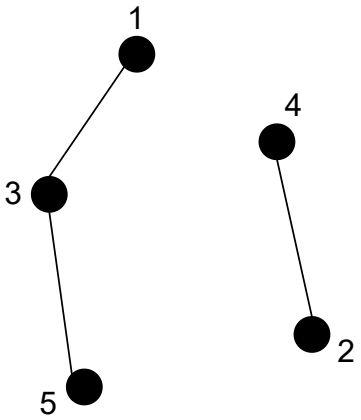


FIGURE 5.11: A path of length 5 between nodes 2 and 6



(A) Connected Graph



(B) Disconnected Graph

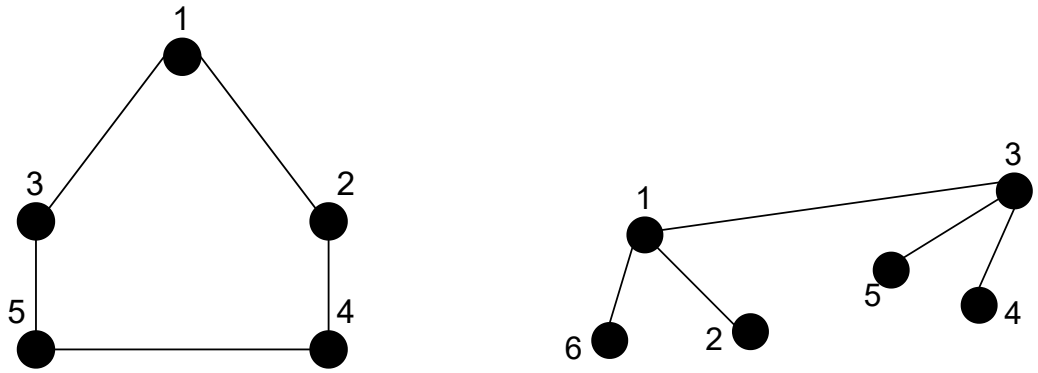
FIGURE 5.12: Comparison between a connected and a disconnected graph

a cycle C is defined as $C = P + x_{k-1}x_0$. If a graph does not contain any cycle, then it is defined as "acyclic". In Figure 5.13 a comparison between a cyclic graph and an acyclic graph is shown.

The set of all paths in a graph, other then defining the connectedness and cyclicity of the graph itself are also at the base of one of the major applications of graph theory, that is, finding the shortest path. The shortest path is defined through the "minimum spanning tree", that is a sub-graph that connects all the nodes of a graph with either the least number of hops or least weighted paths. In what follows, the most commonly used techniques used for finding the shortest paths on a graph are discussed.

5.3.2 Shortest Path Algorithms

In graph theory, the shortest path problem is the problem of finding the shortest route between two vertices (or nodes) in a graph. In the case of unweighted graphs, generally the shortest path between two nodes A and B corresponds to the path characterized by the lowest number of nodes between the two nodes. On the other hand, the shortest path between two



(A) Cyclic Graph (B) Acyclic Graph

FIGURE 5.13: Comparison between a cyclic and an acyclic graph

nodes A and B on a weighted graph corresponds to the path with the minimum sum of the weights of its constituent edges [118], [119].

The shortest path problem is general enough to be defined for graphs whether undirected, directed, or mixed. In this section the problem is defined for undirected graphs, anyway for directed graphs the definition of the problem is identical with the exception that the concept of path requires that consecutive vertices are connected by an appropriate directed edge.

A path in an undirected graph is a sequence of vertices $P = (v_1, v_2, ..., v_n) \in V \times V \times \cdots \times V$ such that v_i is adjacent to v_{i+1} for $1 \leq i < n$. Such a path P is called a path of length $n - 1$ from v_1 to v_n. Note that v_i are variables; their numbering relates to their position in the sequence and does not need to necessarily relate to a canonical labeling of the vertices.

Let $e_{i,j}$ be the edge incident to both v_i and v_j. Given a real-valued weight function $f : E \to \mathbb{R}$, and an undirected (simple) graph G, the shortest path from v to v' is the path $P = (v_1, v_2, \ldots, v_n)$ (where $v_1 = v$ and $v_n = v'$) that over all possible n minimizes the sum $\sum_{i=1}^{n-1} f(e_{i,i+1})$. When each edge in the graph has unit weight, that is, $f : E \to \{1\}$, (or alternatively if we are considering unweighted graphs) this is equivalent to finding the path with fewest edges.

The above stated problem is also often defined as "single-pair shortest path problem" [120]. This definition distinguishes the classical shortest path problem from some of its variations:

- The single-source shortest path problem, in which we have to find shortest paths from a source vertex v to all other vertices in the graph;
- The single-destination shortest path problem, in which we have to find shortest paths from all vertices in the directed graph to a single destination vertex v. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph;
- The all-pairs shortest path problem, in which we have to find shortest paths between every pair of vertices v, v' in the graph.

In the context of this work, for reasons of brevity the shortest path problem will refer to the single-pair shortest path problem.

The most common algorithms used for solving this problem are [118]:

- Dijkstra's algorithm solves the single-source shortest path problem with non-negative edge weight;
- Bellman–Ford algorithm solves the single-source problem if edge weights may be negative;
- A search algorithm solves for single-pair shortest path using specific heuristics to try to speed up the search;
- Floyd–Warshall algorithm solves all pairs shortest paths.

In the context of this work it is particularly important to focus on the Dijkstra's algorithm, which is also one of the most popular solutions to the shortest path problem.

In what follows a brief description of the Dijkstra's solution is detailed.

Given a graph G, let the node at which we are starting be called the "initial node". Given a node Y from G, let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them at each iteration of the algorithm.

1. Mark all nodes as "unvisited". Create a set of all the unvisited nodes, this set is defined as "unvisited set";
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other unvisited nodes. Set the initial node as current;
3. For the current node, consider all of its unvisited neighbours and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node is A, and A is marked with a distance of 6, and the edge connecting it with a neighbour B has length 2, then the distance to B through A will be $6 + 2 = 8$. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept;
4. When all of the unvisited neighbours of the current node have been considered, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again;
5. If the destination node has been marked as "visited" or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished;

6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

When planning a route, it is not necessary to wait until the destination node is marked “visited” as described above. In fact, the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

Chapter 6

Active Directory

In this chapter, Active Directory and its security concepts are discussed. Active Directory is the core of enterprise networks nowadays, hence, understanding its foundations is particularly important from a security perspective. First, basic concepts about the Windows operating system security are highlighted, then the protocols and technologies used in Active Directory environments as well as the security attacks affecting these environments are detailed.

6.1 Microsoft Windows Security

Understanding notions related to the Microsoft operating systems is particularly important, because it is frequently employed as the main operating system for workstations and domain controllers in enterprise organizations. Moreover, Active Directory environments are primarily designed, supported and documented for the Windows Operating System. These operating systems are proprietary and complex enough that a complete description of the security aspects would require an entire book, it is therefore out of the scope of this work to provide a comprehensive documentation about all security aspects. Nonetheless, a description of the fundamental building blocks of Windows security are provided in what follows. More precisely, the concept of SID (Security Identifier) will be detailed with its implications. The role of Default Service Accounts used as security principals is discussed and general considerations about Windows access control mechanisms are provided. In addition concepts involved with common privilege escalation attacks are covered, such as the mandatory integrity control, access token duplication, privileges and security descriptors.

6.1.1 SID

SID stands for Security Identifier and is a unique identifier for objects to which it is possible to assign permissions or control access. The entities which own a SID are called "Security Principals" and generally correspond to users, groups or services. Basically every object having a SID can be managed from a security perspective.

A SID is identified by a unique string, (e.g., in the form `S-1-5-21-1692`), and can be decomposed into two parts:

- the entire string before the last dash is known as the "root SID"(i.e., `S-1-5-21-` from the example above) and identifies the computer and the domain;
- the string after the last dash is known as the "RID" (i.e., `1692` from the above example) and corresponds to a relative identifier that is specific for a user or group; RIDs differentiates specific users within a computer or a domain.

Notice that Windows defines some built-in SIDs for default entities that are present on every installation of the operating system. Some of these are worth to mention for their importance:

- `S-1-1-0: Everyone`, represents all users;
- `S-1-2-0: Local`, represents all users who log on physically;
- `S-1-5-18: Local System`, represents a special service account;
- `S-1-5-20: Network Service`, represents a special service account;
- `S-1-5-19: Local Service`, represents a special service account;
- `S-1-5-32-544: Administrators`, the administrative group;

The SID is a very important concept in Windows environments and it is worth to remind that everything that does not have a SID cannot be controlled from a security perspective since no access control can be imposed.

6.1.2 Default Service Accounts

Within the Microsoft Windows operating systems accounts can be distinguished into: "User accounts" and "Service accounts". User accounts represent accounts associated to the actual users of the system who are able to log on and are associated to real physical users. On the contrary Service accounts are security principals who cannot log on machines and are associated to services running on a Windows computer. These accounts exist so that Windows can associate privileges and permissions to the running services; this is done to implement the concept of separation of privileges.

On every Windows machine, by default there are three built-in service accounts:

- `NTAUTHORITY\SYSTEM`: also known as "Local System Account". It is a very high-privileged account. It has extensive privileges on the local system and acts as the computer on the network;
- `NTAUTHORITY\LOCALSERVICE`: also known as "Local Service Account" is a built-in account that has the same level of access to resources and objects as members of the `Users` group. This limited access helps safeguard the system if individual services or processes are compromised. Services that run as the Local Service account access network resources

as a null session (i.e., without credentials). Be aware that the Local Service account is not supported for some services such as the SQL Server or SQL Server Agent services;

- `NTAUTHORITY\NETWORKSERVICE`: also known as "Network Service Account" is a built-in account that has an higher level of access to resources and objects with respect to members of the Users group. Services that run as the Network Service account access network resources by using the credentials of the computer account.

Default Service accounts are particularly interesting since they normally run with high privileges and are active by default. For this reason, these accounts represent an interesting target from the perspective of an attacker.

6.1.3 LSASS

LSASS (Local Security Authentication Subsystem Service) is a process in Microsoft Windows operating systems that is responsible for enforcing the security policies on the system. LSASS accomplishes different tasks, such as verifying users logging on to a Windows computer or server, handling password changes or creating access tokens. In addition, this process is also responsible for writing to the Windows Security Log. Let us remark that this process represents one of the major pillars within Windows security.

LSASS is complex and hosts different sub-services used for different purposes, it can be subdivided into these components:

- Active Directory Module: used to interact with Active Directory;
- NetLogon: used to provide secure communications to Active Directory or to other network based resources/services;
- LSA Server: local security policy service which interacts with the LSA database, that is a part of the registry holding information about local users and passwords;
- SAM Server: is the service responsible for the interaction with the SAM database, that is a part of the registry holding additional information about local users, groups and security policies;
- `Msv1_0.dll` and `Kerberos.dll`: responsible for the communication with the domain controllers.

In addition LSASS can interact with “LSAISO” which is a component acting as an additional mitigation mechanism. LSAISO enables a set of security features for Windows. “LSAISO”, when enabled, substitutes LSASS for certain features such as the handling of credentials.

6.1.4 Access Token

Whenever a user logs onto a Microsoft Windows system it will be provided with a key-chain called "Access Token". Windows uses these tokens to ensure that accounts have the right privileges to carry out a particular action. Access tokens are assigned to an account when users physically log in or are authenticated by other means. The assignment of the access token is generally handled by the `LSASS.exe` process. After a login procedure, a user access token typically contains:

- SID of the logging user;
- SIDs of all the groups the user belongs to;
- Privileges, given from user rights assignments, which are rights with respect to some operating systems operations, for example: loading a driver, shutting down the system, reading security logs and privileged tasks;
- Claims (from Windows 8.x/2012), which represent dynamic temporary rights, this is related to a special access control mechanism implemented from Windows Vista.

Note that access tokens are implemented as kernel objects identifying the security context of a user, a process or a thread. We can distinguish between two types of access tokens, primary tokens and impersonation tokens.

Primary tokens are associated with a user account or with a process and are generated when that user logs onto a machine. These tokens are bound to the user for the entire session. On the other hand, impersonation tokens allow a particular process (or thread within a process) to gain access to resources using the token of another user or process, hence performing an impersonation.

Note that, when a user starts a new process, her primary access token is copied and attached to the new process.

By default threads inherit their parent process access token. However threads can impersonate other users by taking advantage of the impersonation mechanisms described. In fact, the process of impersonation takes place anytime a process or thread needs to temporarily run with the security context of another process or user.

Concerning the impersonation token, there are different levels of impersonation that a process can request:

- SecurityAnonymous: current user/client cannot impersonate another user/client;
- SecurityIdentification: current user/client can get the identity and privileges of a client, but cannot impersonate the client;
- SecurityImpersonation: current user/client can impersonate the client's security context on the local system;

- SecurityDelegation: current user/client can impersonate the client's security context on a remote system.

The abuse of these impersonation mechanisms in unintended ways is a very common vector for privilege escalation on Windows systems. In fact, whenever an attacker is able to inject code into a process (for example through buffer overflows), the token impersonation mechanism can be abused to duplicate the access token of the attacked process and spawn a separate process with equal rights.

6.1.5 Access Control Mechanisms

Access control management within Microsoft Windows can be a complex topic. In fact, these operating systems have several security mechanisms designed to manage access control on security principals and resources. More precisely, permissions can be managed through the following components:

- Privileges;
- NTFS Permission Control;
- Mandatory Integrity Control (introduced since Microsoft Windows Vista);
- User Account Control (introduced since Microsoft Windows Vista).

These access control mechanisms have different degrees of priority. In case of conflicts with each other about the access control directives to apply, the one with the highest priority wins. This means that the mechanism having the highest priority imposes its access control directives ignoring the ones imposed by the other mechanisms.

Privileges represent the most powerful form of permission a security principal may have. These are general special permissions which allow users to perform critical actions, such as adding a driver, turning off/on a connected device, mount/unmount a disk and others. Privileges have the highest priority, no other mechanism can be imposed over privileges.

NTFS permissions are related to read/write/access permissions associated to files on NTFS filesystems, which is the most common Windows filesystem type. These type of access control determines if a specific resource can be accessed by a user.

The Mandatory Integrity Control (MIC) is a core security feature starting from Windows Vista and later versions. This feature adds mandatory access control to running processes based on their Integrity Level (IL). The IL represents the level of trustworthiness of an process. The goal of this mechanism is to restrict the access permissions for potentially less trustworthy processes compared with other processes running under the same user account that are more trusted. Basically certain actions may require a specific integrity level, if this integrity level is not met by a process, the operating system prevents the process from performing that action.

The User Account Control (UAC) aims at improving the security of Microsoft Windows by limiting applications to standard user privileges unless

an administrator authorizes an increase or elevation of privileges. In this way only trusted applications can obtain administrative privileges. This allows in some cases to protect the system from malware. In other words, a user account may have administrative privileges assigned to it, but applications that this same user runs do not directly inherit those privileges unless they are approved beforehand or the user has explicitly authorized them.

Let us remark that these mechanisms have many caveats and additional details that have not been discussed because out of the scope of this thesis work. Nonetheless, it is sufficient to remind that Windows associates different priorities to these access control mechanisms and in particular, it is important to remember that the access control mechanism characterized by the highest level of priority corresponds to the "privileges". In fact, privileges can bypass all other access control mechanisms.

In what follows, a more detailed description of the most relevant access control mechanisms is provided.

6.1.6 Privileges

Privileges are user rights attachable to the access token who allow users to perform privileged "special" actions. Examples of these actions include: loading a driver, shutting down the system, reading security logs, changing time/zone settings. As already discussed, privileges correspond to the highest priority access control mechanisms and privileges can all other constraints imposed or restrictions. Privilege names start with "`Se`" and end with "`Privilege`". Privileges are managed by the so called attached to users on by using the local policies manager (i.e., `gpedit.exe`). Notice that "User Rights Assignment" contains both privileges and other user rights that do not qualify as privileges. These other user rights are simply called "User Rights" and differ from privileges since they are not attached to the access token. This happens because "User Rights" refer to pre-login activities, hence, they cannot be attached to the access token (remember that privileges and in general the access token is provided only after login. This means that these "User Rights" are taken care of by the Security Reference Monitor (SRM) to be enforced.

Administrators generally have a significant amount of privileges, among these some interesting ones are:

- Ownership: which allows access to any resource;
- Debug: allows adjusting processes access tokens and manipulate memory;
- Backup and Restore: these are two different privileges allowing in the case of "backup" to read all data and in the case of "restore" to write to any location;
- Impersonate: allows an admin to impersonate any logged on user.

Let us remark that the privileges associated to an account (provided to the account when it is created or inherited from a group) may allow a user to carry out sensible actions. Since privileges are commonly abused by attackers to perform privilege escalation attacks, in what follows are reported the most commonly abused privileges:

- `SeImpersonatePrivilege`: grants the ability to impersonate any access tokens which it can obtain. The infamous "Juicy Potato" uses this privilege;
- `SeAssignPrimaryPrivilege`: similar to the SeImpersonatePrivilege and enables a user to assign an access token to a new process. Again, this can be exploited with the "Juicy Potato" exploit;
- `SeBackupPrivilege`: grants read access to all objects on the system, regardless of their ACL. Using this privilege, a user could gain access to sensitive files or extract hashes from the registry that can be cracked or used in a pass-the-hash attack;
- `SeRestorePrivilege`: grants write access to all objects on the system, regardless of their ACL. There are multiple techniques to abuse this privilege, for example by modifying service binaries, overwriting DLLs used by SYSTEM processes, modifying registry settings and others;
- `SeTakeOwnershipPrivilege`: allows the user take ownership over an object (this is the `WRITE_OWNER` permission). Once we own an object we can modify its ACLs and grant write access to ourselves. At this point, the same methods used with SeRestorePrivilege apply;
- `SeCreateTokenPrivilege`: allows the user to create another access token;
- `SeLoadDriverPrivilege`: allows a user to load a device driver, thus, executing low level privileged code;
- `SeDebugPrivilege`: allows a user to manipulate the memory.

Note that it is useless to implement group policies aimed at limiting the privileges of an Administrator, since Administrators can always bypass group policies.

6.1.7 Mandatory Integrity Control

Microsoft Windows Vista introduced MIC (Mandatory Integrity Control) which made it possible to differentiate a user's processes according to a level of trustworthiness. Mandatory integrity level control, differs from the classical permission based access control since it takes into account the application we are using and how much we can trust this application. MIC indeed, doesn't care "who" is doing something but "what" is being used.

Whenever a resource residing on a Windows system has to be accessed, different types of access control mechanisms are considered. Each of these

mechanisms checks if there are appropriate rights to successfully perform an operation on an object. As already described these mechanisms have different priorities.

In terms of priorities MIC represents the strongest access control mechanism after the evaluation of Privileges. In fact, MIC permissions are very powerful. For example, even if we have the sufficient NTFS permissions to access a resource we may not access it if we do not have a sufficient mandatory level to access that resource.

MIC integrity levels are subdivided into:

- `SYSTEM`, highest MIC level;
- `HIGH`, used when we explicitly run processes as an Administrator (e.g., through the "Run as Administrator" pop-up);
- `MEDIUM`, Default MIC level;
- `LOW`, used by insecure applications such as IE and sometimes referred to as "AppContainer".

A MIC is also stored within the access token associated to each user. By default, running as a standard user sets the integrity level to `MEDIUM`, while running as an administrator sets the integrity level to `HIGH`. Services running under one of the three default service accounts (Local System, Network Service, Local Service) have integrity level set to `SYSTEM`, that is, the highest. In addition all processes and resources are assigned with an integrity level, and interaction is possible only when the integrity level of a process is at least equal or higher to the one of the resource. Note that, to further separate processes from sensible resources we can observe that modern Microsoft Windows versions have three directories in the user profile:

- `Roaming`: containing data that is synchronized over AD;
- `Local`: containing a running environment for applications running with a `MEDIUM` integrity level;
- `LocalLow`: containing a running environment for applications running with a `LOW` integrity level.

For example, Internet Explorer normally runs with a `LOW` integrity level, but when we have to download something, a new window pops up. This is done in order to spawn a different process running with a `MEDIUM` integrity level.

In addition to the MIC, from Microsoft Windows 8 to further enhance the concept of separation of privileges also "Modern Apps" have been introduced. These "apps" are also known as "Universal Apps" and are programs containerized to be isolated from the rest of the system. Each one of these applications have:

- a SID associated with the application;

- additional SIDs to enhance its capabilities;
- a dedicated, per-user specific separate directory;
- a separate object manager namespace;
- a separate installation directory for the application.

Moreover, to enhance the notion of MIC, Microsoft recently also introduced the concept of "protected processes". These are processes designed to create an additional barrier used to protect processes not only from user processes but even from administrators. These mechanisms are typically found in antivirus solutions.

6.1.8 Security Descriptor

A security descriptor contains the security information associated with a securable object. Basically this is analogous to access tokens, but while access tokens is associated to users and processes, security descriptors are related to resource objects such as files or devices. Each resource upon creation is provided with a security descriptor. Each security descriptor is composed by different entries:

- Owner SID, ownership information about an object;
- DACL (Discretionary Access Control List): a list describing what permissions specific security principals have on a specific object;
- SACL (System Access Control List): a list specifying auditing and tracing directives. In this list information such as who we are auditing and against what is provided. SACL allows to determine which operations by which users should be logged in the security audit log.

Each of the access control lists described contains zero or more entries known as access control entries (ACE). Each ACE contains a SID and an access mask.

Note that the order in which a DACL is specified is important and plays a fundamental role in understanding how access control is applied to an object. In fact, the order of application of DACL entries follows a policy that makes the higher (in the list) more important with respect to the lower ones. For example, considering a user named Joe belonging to two groups, namely, "Administrators" and "Writers", if Joe tries to write to an object X which has the following DACL:

- Allow Bob Write;
- Deny "Writers" Read/Write;
- Allow Joe Read/Write.

Joe is denied to access the object X with write permissions, because the second ACE specifies that "Writers" cannot read or write to the resource and Joe belongs to this group. Note that, although the third ACE explicitly specifies that Joe can access the resource, this is ignored, because the second ACE is more important (higher in the list) with respect to the third ACE. Basically ACLs evaluated first have higher priority. On the other hand, if we swap the last two entries, then Joe would be able to write to X. Notice that if there are no ACEs related to Joe, then he would be denied by default to access the object. Contrarily, if no DACL is specified at all, by default the access is granted to everyone.

6.2 Active Directory

Active Directory (AD) is a set of technologies developed by Microsoft implementing directory services with the aim of managing complex computer networks. This technology was originally built-on top of Windows 2000, and has evolved over years through multiple Windows releases. Active Directory is implemented through a client/server architecture and enables network administrators to manage information, users, computers, groups, services and more in general network policies. This administration is efficiently performed through a central repository. Basically, once a resource (e.g., a user, a computer, a printer) has been added to the Active Directory infrastructure, it can be made available for the entire managed enterprise or to a subset of selected users. Let us remark that the structure of the information stored within directory services very often matches the structure of the organization that is using it. In what follows the logical structure of Active Directory is described.

6.2.1 Logical Structure

From a bird's eye view, Active Directory systems are logically organized into domains, trees and forests.

An Active Directory domain is a container for collections of objects (e.g., users, computers). Each domain is identified by a namespace that corresponds to a DNS domain name and holds a database containing objects and corresponding identity information. Domains can be related to other domains. Whenever one or more domains share a contiguous namespace, a tree of domains is composed. The set of all related domains under a common namespace is defined as a forest. A forest is a collection of trees that share a search engine for AD objects (also known as "global catalog"), a directory scheme, a logical structure, and a directory configuration. A forest is at the top of the Active Directory hierarchical structure and it represents the security zone where users, computers, groups, and other resources are accessible. In fact, elements within a forest cannot communicate with elements within other forests unless a trust relationships between these forests is explicitly defined. Figure 6.1 shows an example of an Active Directory forest consisting of seven domains (represented by triangles) organized into two trees. As

can be seen, the domains belonging to the same tree (e.g., `testlab.com` and `europe.testlab.com`) are linked by a parent-child relationship. On the contrary, domains belonging to different trees of the same forest (e.g., `testlab.com` and `caronte.com`) are not linked by any relationship.

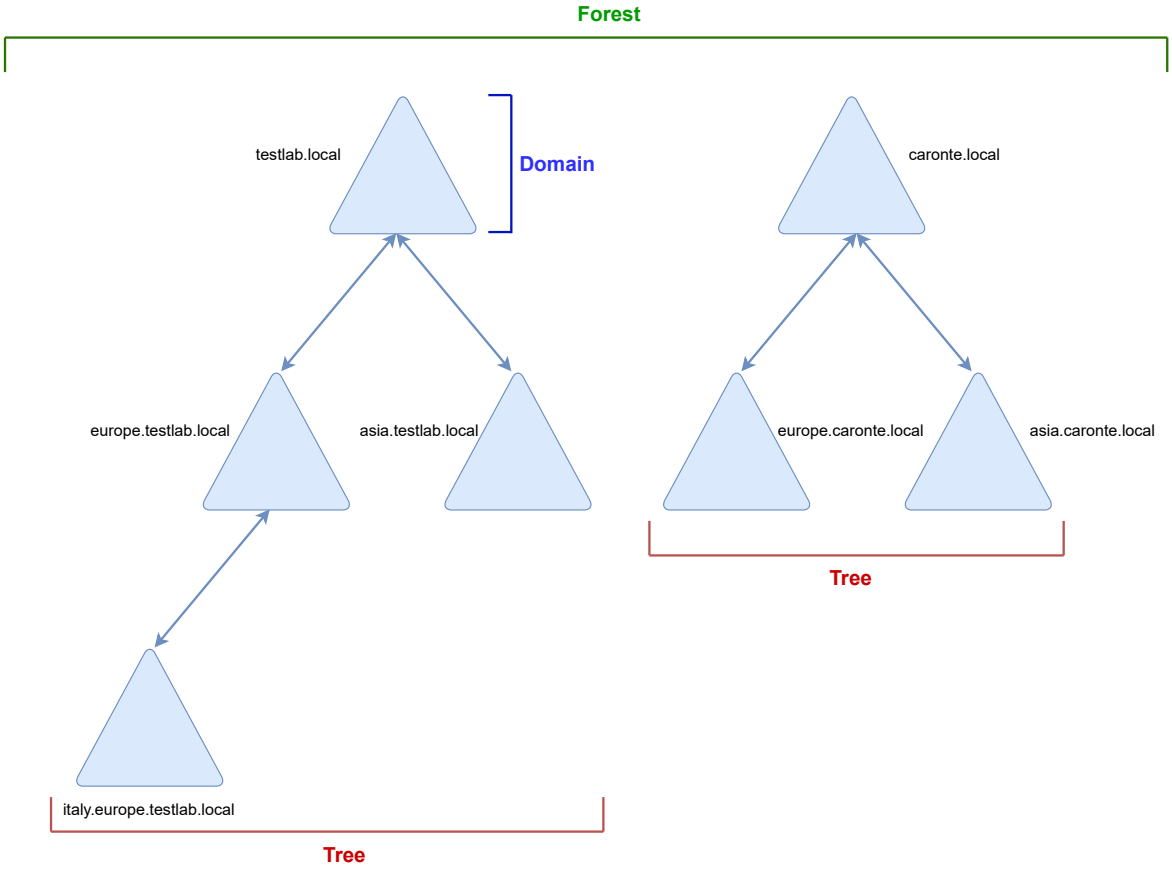


FIGURE 6.1: Example of Active Directory forest

In addition to the forest trust relationships, system administrators can define domain trust relationships. A domain trust relationship is a relationship between two domains that allows the users of a domain to be recognized by the Domain Controllers of the other domain. In this way, authorized users can access objects belonging to the trusted domain. As shown in Figure 6.2, there are four main types of domain trusts:

- Two-way: this is established by default in parent-child domain relationships. More precisely, a new child domain automatically trusts the parent domain and vice versa;
- Transitive: when domain *A* and domain *B* are linked by a parent-child relationship, and domain *B* and *C* are also in a parent-child relationship. In this case domain *A* and domain *C* trust each other in an implicit way;

- Shortcut Trust: when domain *B* and domain *D* belong to the same forest and are non-adjacent (i.e., they are not linked by a two-way-parent-child relationship), it is possible to define a one-way transitive relationship between *B* and *D* to reduce the time needed for computing and traversing a trust path;
- External Trust: it is a one-way-non-transitive relationship between two domains belonging to two different Active Directory forests. Thus, it is necessary to combine two one-way relationships in order to generate a two-way trust relationship.

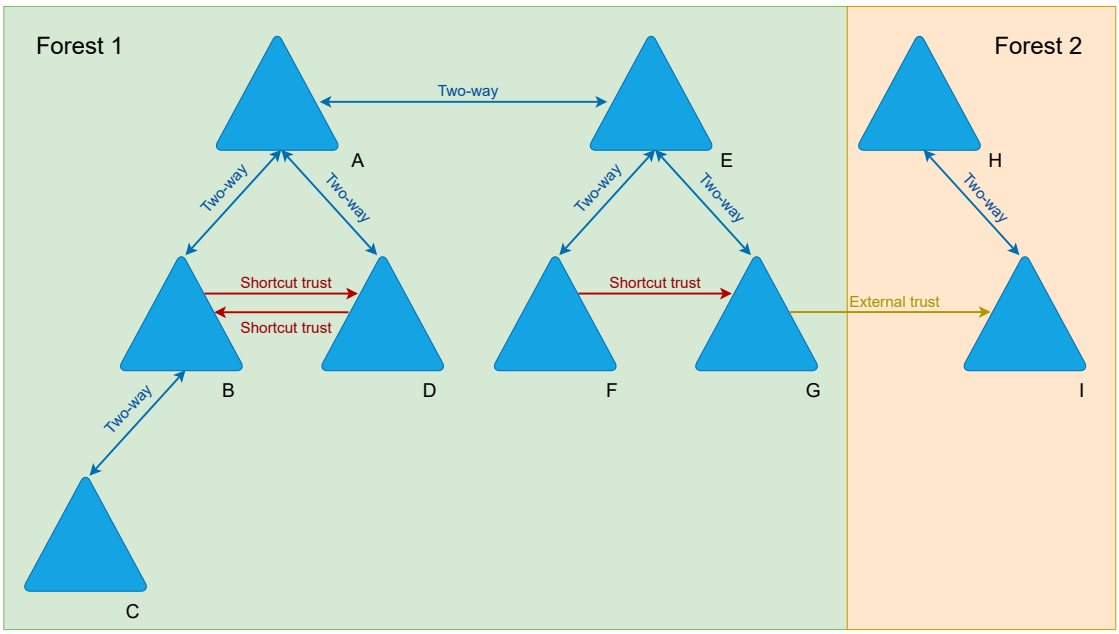


FIGURE 6.2: Examples of Active Directory domain trust relationships

6.2.2 Active Directory Data and Objects

The main repository used by Active Directory to store all the information is a distributed database organized into three partitions, respectively: Schema, Configuration and Domain. Each partition contains specific type of data and follows a specific replication pattern. Microsoft often refers to these partitions as 'naming contexts'. In detail, the 'Schema' partition contains the definition of object classes and attributes within the entire Forest. The 'Configuration' partition contains information on the physical structure and topology of the sites where the environment is installed. Finally, the 'Domain' partition holds all objects created within a specific domain. Basically these partitions store mainly Active Directory objects, it is then important to understand what are these objects.

In fact, "objects" represent a fundamental concept in Active Directory. An object represents a resource on the network, that typically is a user, a group, a computer, an application, a printer, or a shared folder. Objects are stored in the Active Directory database within three categories, namely:

- Security Principals, i.e., users, computers, and groups;
- Resources, such as printers, scanners;
- Services, such as email service or file share service.

Anyway for the scope of this work, we can generalize and group objects only using two categories. In fact, from a security point of view, objects fall into two broad categories: "resources" (e.g., printers, file shares) and "security principals" that is everything that can be controlled from a security perspective as described in 6.1. Every object is uniquely identified by what is known as the Global Universal Identifier (GUID) and security principals also have an additional identifier used for security purposes known as Security Identifier (SID). Hence, every object has a GUID, but only security principals have a SID. Note that GUID for an object is practically unique in the whole world and never changes. On the contrary, unlike the GUID, the value of the SID can change. For example, when users are moved to another domain of the forest, their SIDs will change, but their GUIDs will remain unchanged. Let us remark that each object is defined and described also by a set of attributes. For example, each user in Active Directory other than having a unique GUID and a unique SID, is described by a set of additional descriptive attributes. Examples of these attributes are name, e-mail address or her/his phone number. All objects of the same type (or class) have the same set of attributes. Some attributes are required to have values (e.g., the first name attribute of a user object), while other attributes can be optional (e.g., Phone Number). The RID, that is, the last part of the SID, uniquely identifies a security principal relative to the local or domain security authority that issued the SID. Any group or user that has not been created explicitly has a RID of 1000 or greater by default; this is in fact true for all built-in accounts. In fact, by default, the value of the RID of every Security Principal created by an administrator is greater or equal to 1,000, while the values of the SID of default Security Principals generated by Microsoft Windows are lower than 1,000. The unique identifiers associated to each object and all the attributes described are defined in the Active Directory Schema. In this schema a class is a component of the Schema that defines the type of an object and the set of its mandatory and optional properties. In fact, every object stored in the schema is an instance of a specific object class.

The Schema definitions are objects themselves that are stored in the database as types *Class Schema* or *Attribute Schema*. Objects that are logically related can be grouped into containers, these containers are known as organizational units (OUs). These units are container objects used to organize other objects. OUs provide hierarchy to a domain, simplify its administration and often resemble the organization's structure in managerial or geographical terms. OUs can contain other OUs, they act as directories. The OU is the recommended level at which to apply group policies, which are Active Directory objects formally named group policy objects (GPOs), although policies can also be applied to the entire domain. The OU is the level at which administrative powers are commonly delegated. The objects are generally grouped into

Organizational Units and OUs according to geographical, logistic or management needs. Organizational Units are directory objects whose sole purpose is containing other objects and child Organizational Units. Organizational Units are an abstraction used to facilitate the logically organization of objects and the management of Group Policies and administrative privileges. They are identified using a unique identifier like any other Active Directory object.

Let us remark that SIDs play an important role in Access Control Lists as described in 6.1. In fact, Microsoft Windows NT uses SIDs to identify the objects with rights on other Active Directory objects.

In fact, every security principal has a SID. Moreover it is important to note that security principals themselves can be divided into four distinct categories within Active Directory environments:

- Domain Admins, these are the users who have administrative rights over the domain and are the only ones with access to the domain controller;
- Service Accounts, (that can also be domain admins) these are never used by physical users but are used by Windows to pair services with an account to which access control policies can be applied;
- Local Admins, these are the users who have administrative rights over a specific machine but not on the domain;
- Domain Users, users who can log onto the domain.

Objects within an AD environment are replicated following different strategies among other machines. This is done in order to improve the availability of the services provided by Active Directory. In particular the "Schema" and the "Configuration" partitions replicate to all domains within the Forest, while the "Domain" partition replicates only within its domain. Note that replication of Active Directory relies on Remote Procedure Calls (RPC) over IP (RPC/IP), a protocol described in the next section.

6.2.3 Protocols

In this section a brief description of the main protocols involved within Active Directory environments is provided. Let us remark that Active Directory relies on many different protocols and technologies and this is also the reason behind its complexity. For each of these protocols the main features together with the main security threats are discussed.

DNS

DNS (Domain Name System) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. Its main task is related to translating each human friendly names to IP addresses of computers. DNS protocol uses UDP port 53 for

most of its functions. The protocol is described in RFC 1034 [121] and RFC 1035 [122].

Active Directory Domain Services (AD DS) relies on DNS to discover the domain controllers hosting the directory services. In this sense, DNS is needed by all computers in a domain for performing basic Active Directory operations, such as search queries, authenticate users, or updates.

In Active Directory, each DNS server must provide support for Service Location (SRV) resource records, as described in RFC 2052 [123]. SRV resource records in this context translate the name of a service to the IP address of the domain controller that offers that particular service. Furthermore, SRV resource records allow domain administrators to use several servers for hosting domain services and to designate some servers as primary and other as backup servers. It is also recommended that each DNS server provides support for DNS dynamic updates as specified in RFC 2136 [124]. In this way, it is possible to improve the DNS administration by reducing the time needed to manually manage zone records.

The most useful and common SRV records used within Active Directory domains are:

- `_gc._tcp`: enable a client to find a Global Catalog server;
- `_ldap._tcp`: enable a client to find an LDAP server;
- `_kerberos._tcp`: enable a client to find an KDC (Key Distribution Center) server;
- `_kpasswd._tcp`: enable a client to locate a server that is running the Kerberos Password Change service over TCP.

Active Directory provides a built-in method of storing and replicating DNS records within a defined set of machines, called "zone".

All of the records stored within the zone are replicated to other DNS servers by using AD replication services. In practice, each domain controller stores a writable copy of the DNS zone data for namespaces for which they are authoritative. Active Directory zones also provide the ability to use secure dynamic updates, which supports controlling which computers may make updates and prevents unauthorized changes from being made.

DNS zone data is stored in an application directory partition. A forest-wide partition named `ForestDnsZones` is used for the zone data. For each AD DS domain, a domain partition named DomainDnsZones is created. Typically, DNS implementations are used with a contiguous namespace.

DNS: Security Threats

To mitigate the threats associated with the DNS protocol, DNS zones can be secured by using secure dynamic updates, restricting zone transfers, plus implementing zone delegation and DNS Security Extensions (DNSSEC). By using secure dynamic updates, computers will be authenticated through Active Directory, and security settings will be applied when performing a zone transfer.

Additionally, zone transfers can also be restricted to specific IP addresses within the network. Zone delegation can be approached by using two methods.

The first method involves the constraint of delegating DNS changes to a single team or entity. Hence all changes must be tracked and approved. This method limits the amount of people making changes but also creates a single point of failure. The second method involves delegating zones to multiple teams which will be managing each component of a network or domain. While changes may still need to be approved and tracked, this spreads out the risk among multiple points of failures. Hence, this second method may limit damage if a single component is compromised.

Although there are not types of DNS attacks that are specific to Active Directory, these directory services are susceptible to the common threats to which the DNS is exposed. The most relevant threats to take into account are related to denial of service and poisoning, more precisely:

- Distributed Denial of Service (DDoS): it consists of sending many DNS requests to the target DNS server (in the specific case to the domain controller that works also as a DNS server) in order to make it unavailable. Generally, a DDoS attack is carried out using a network of computers called botnet;

- Source IP spoofing DDoS: DNS protocol relies on UDP; thus it is very easy to spoof the source IP address. In this way, it is possible to overcome the DDoS mitigation technique that consists of blocking an IP address after a specific number of requests;

- Reflected DDoS: it consists of spoofing the source and the destination IP addresses, so that, querying a set of legitimate DNS servers, it is possible to flood the victim DNS server with a large amount of DNS responses;

- Cache Poisoning: this attack consists in changing the DNS responses stored in the DNS cache so that any user that queries the DNS server will obtain a poisoned response from the cache and will be redirected to a malicious site;

- Unauthorized Zone Transfer: this attack allows an attacker to get full information about all the subdomains available for a specific domain. This is generally due to an access control misconfiguration allowing non-authorized users or non-privileged users to perform a DNS zone transfer, revealing the whole attack surface related to a domain.

LDAP

The Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network as. LDAP latest specifications are defined in the list of RFCs going from

RFC4511 to RFC4519 [125–133]. LDAP is a binary protocol, but in general a common format used to represent LDAP entries is provided by the LDAP Data Interchange Format (LDIF) as described in RFC2849 [134].

A common use of LDAP is to provide a central place to store usernames and passwords. This allows different applications and services to connect to the LDAP server to validate users. A client starts an LDAP session by contacting an LDAP server, called a Directory System Agent (DSA), by default on TCP and UDP port 389, or on port 636 for using LDAP over SSL (LDAPS). In Active Directory LDAP is additionally used for global catalogs over ports 3268 and 3269 (LDAPS). The client then sends an operation request to the server, and the server sends responses in return. For most types of LDAP queries, there is no need for the client to wait for a response response before sending the next request, and the server may also send the responses in any order. All information is transmitted using Basic Encoding Rules (BER) [135].

The client may request the following operations to an LDAP server:

- StartTLS, use the LDAPv3 Transport Layer Security (TLS) extension for a secure connection;
- Bind, authenticate and specify LDAP protocol version;
- Search, perform a search for and/or retrieve directory entries;
- Compare, test if a named entry contains a given attribute value;
- Add a new entry;
- Delete an entry;
- Modify an entry;
- Modify Distinguished Name (DN), that corresponds to moving or renaming an entry;
- Abandon, abort a previous request;
- Extended Operation, generic operation used to define other operations;
- Unbind, close the connection.

In addition the server may send "Unsolicited Notifications" that are not responses to any request, e.g. before the connection is timed out.

The protocol provides an interface with directories that follow the 1993 edition of the X.500 model where an entry consists of a set of attributes. An attribute has a name (an attribute type or attribute description) and one or more values. The attributes are defined in a schema (see below). Each entry has a unique identifier: its Distinguished Name (DN). This consists of its Relative Distinguished Name (RDN), constructed from some attributes related to the entry, followed by the parent entry's DN.

The DN can be thought of as a full file path and the RDN as its relative filename in its parent folder (e.g. if `/foo/bar/myfile.txt` is the DN, then `myfile.txt` is the corresponding RDN).

A DN may change over the lifetime of the entry, for instance, when entries are moved within a tree. To reliably and unambiguously identify entries, a UUID might be provided in the set of the entry's operational attributes.

An example of LDAP entry may be:

```
dn: uid=user,ou=people,dc=example,dc=com
changetype: add
objectClass:top
objectClass:person
uid: user
sn: last-name
cn: common-name
```

Note that all LDAPv3 servers should provide a special LDAP entry that provides information about the capabilities of that specific server and the data that it contains. This entry is called the rootDSE, where DSE stands for DSA-Specific Entry, and it is a special entry characterized by a null DN (i.e., the DN with zero RDNs, and a string representation that is the empty string).

LDAP: Security Threats

LDAP allows authentication via SASL mechanism as specified in RFC4422 [136]. As LDAP includes native anonymous and name/password (plain text) authentication methods, the ANONYMOUS authentication specified in RFC4505 [137] and PLAIN SASL mechanisms are typically not used with LDAP.

The main threats to an LDAP directory service include (but are not limited to) access control misconfiguration, spoofing and denial of service. Access control should be properly configured by system administrators, common threats related to the authorization processes are:

- Unauthorized access to directory data via data-retrieval operations due to misconfigurations in the access control mechanisms;
- Unauthorized access to directory data by monitoring access of others, by taking advantage of Man-In-The-Middle (MITM) techniques;
- Unauthorized modification of directory data;
- Unauthorized modification of configuration information.

Spoofing threats can instead lead to:

- Tricking a user (i.e., client) into believing that information came from the directory server when in fact it did not;
- Tricking a user (i.e., client) into sending privileged information to a hostile entity that appears to be the directory server but is not;
- Tricking a directory server into believing that information came from a particular client when in fact it came from a hostile client.

In addition to these threats LDAP can also be vulnerable to Denial of Service (DoS) attacks, where the aim is denying service availability to other users by making an excessive use of the resources of the directory server.

Kerberos

Kerberos is a network protocol used for authentication that uses tickets as a form of communication RFC 4120 [138]. The protocol originated at MIT but over time different implementations have been popularized. Its name is inspired by the Greek mythology three-headed dog, Cerberus. In fact, Kerberos is based on the interaction of three parties: the client, the resource server and the Key Distribution Center (KDC).

Older implementations of Kerberos used UDP port 88 by default, but modern implementations must support both TCP and UDP communications over port 88. Active Directory makes use of both ports and in addition, provides a Kerberos password management service on ports 464 TCP and 464 UDP.

Active Directory implements its authentication and Single Sign On capabilities through a Microsoft implementation of Kerberos v5. This implementation slightly differs from the original Kerberos v5 RFC specification and the main difference is that Microsoft Kerberos adds authorization capabilities in addition to the pre-existing authentication features of the protocol.

The Microsoft Kerberos implementation (also dubbed as NT Kerberos) is the most famous and common Kerberos implementation existing nowadays because of Active Directory. In fact, the only time we see Kerberos outside the Microsoft world is when providing Single Sign On capabilities to GNU/Linux servers. Windows 2000 and later versions use Kerberos as its default authentication method in Active directory. Some Microsoft additions to the Kerberos protocol are documented in RFC 3244 [139]. Within Active Directory, Kerberos is used as preferred authentication method: in general, joining a client to a Windows domain means enabling Kerberos as the default protocol for authentications from that client to all the services in the Windows domain and all domains with trust relationships to that domain. In contrast, when either client or server or both are not joined to a domain (or not part of the same trusted domain environment), Windows will instead use NTLM for performing authentication between a client and a server.

Kerberos works in what is in its lingo called "Kerberos Realm", this basically corresponds to the domain name in Active Directory environments. In fact, Kerberos realms are named after domain names, e.g., `MYDOMAINNAME.ORG`. Note that realms, differently from domains, must always be specified in the uppercase notation.

Within a Kerberos realm, there are three types of parties:

- Users: identified by User Principal Names (UPNs) which must authenticate with the KDC to interact with the services;
- Services: identified by Service Principal Names (SPNs), which are basically the services available in a realm;
- KDC (Key Distribution Center): that is the Kerberos server, this corresponds to the domain controller in Active Directory. The KDC authenticates user with a component known as "Authentication Server" and provides tickets which are used by users to authenticate to services

available on the domain with a component known as "Ticket Granting Service".

For example, if a user `jdoe` has a domain account in the `EXAMPLE.COM` Kerberos realm (i.e., that is corresponding to the domain name `example.com`), then this user will have a UPN corresponding to it denoted as `jdoe@EXAMPLE.COM`. At the same time, if this same user initiates a connection to the share path `\\server1.example.com\SharedDirectory` then its workstation will first lookup the computer `server1.example.com` in the Active Directory database and then read its SPN attribute (cifs/server1.example.com). Note that `cifs` indicates a file sharing service type. Remember that AD domains are an integrated system of DNS, LDAP, Kerberos and other services, so nowadays Kerberos comes embedded in AD environments and Domain Controllers act as Key Distribution Centers. Under the hood, domain names are converted to equivalent Kerberos principal names, which have a similar format and are represented in the form `fred@SERVER.EXAMPLE.COM`. The Kerberos realm name is always case-sensitive and by convention always uppercase. Each Active Directory domain acts as a Kerberos realm, and has exactly one realm name (even if multiple UPN suffixes are configured). Every AD domain controller also acts as a Kerberos KDC for the corresponding Kerberos realm.

Note that in Active Directory, Kerberos requires valid DNS names, in the case where IP addresses are provided then authentication will fall back to NTLM (e.g., requesting `\\10.10.10.10\share` will use NTLM for authentication).

Individually, the terms "domain" and "realm" in Active Directory have the same meaning, but are used for different purposes. Realms and realm names come from the Kerberos authentication protocol, where they serve practically the same purpose as domains and domain names. They have no direct relation, strictly speaking, but in practice nearly all Kerberos realms are named after the corresponding DNS domain. The Authentication Service is the first point of contact the client has with the Kerberos system. Basically, this service is used to lookup the user's password and create the Ticket Granting Ticket (TGT). The AS also creates the session key that the user will use for future communication with Kerberos.

The Ticket Granting Ticket (TGT) is the Kerberos ticket for the Ticket Granting Service which runs on the KDC. This ticket is encrypted using a key called "KDC key" belonging to the `KRBTGT` domain Kerberos account and signed using the requesting user password hash. This means that only a KDC can decrypt and read the TGT tickets. TGT have a default expiration of 10 hours within Active Directory systems, but it can be renewed by default for a maximum of 7 days of usage.

TGT Tickets obtained by users are sent to the KDC to obtain Service Tickets for the Ticket Granting Service (TGS) on the KDC. A client requests a ticket by performing a requesting a TGT (Ticket Granting Ticket) along with credentials. The server at this point will check the credentials and if these are correct will send back a TGS (Ticket Granting Service) which comprehends a secret key which will be stored on the client. Now the client will have this ticket until it expires. Now let's say that there are on this networks

other hosts/servers with available services. Services in Kerberos environments have what we call “SPN” or Service Principle Names, these can be SQL servers, antivirus or really any other kind of service we may come up with. At this point if the client wants to connect to one of these services, let’s say an SQL database service, he has to know the SQL database SPN, and then has to send a request containing among other information the SPN which he wants to use in the form of a TGT request. At this point the server will reply with a TGS containing the session key to the SPN which will allow the client to connect. Let us remark that the possession of a valid TGT allows the request for a TGS ticket corresponding to a specific SPN.

The Kerberos authentication process

The steps that take place when a user logged inside a domain wants to access a specific application server (i.e., service) within the same domain are the following:

- A user sends a timestamp encrypted with its NTLM hash and sends it to the KDC, this kind of request is called `AS-REQ`. Note that this step of encryption is skipped if the user has pre-authentication disabled. In this case, in fact, the user can just request a TGT;
- At this point since the DC/KDC has access to all the secrets in the domain it has the password of the user and can decrypt this request and verify that the `AS-REQ` request was actually sent by a valid user. At this point the KDC responds with a TGT; this TGT ticket is encrypted and signed with the NTLM hash of a special account belonging to the domain controller called `krbtgt`. Notice that this special account is used specifically for this purpose. This kind of response is called `AS-REP`. Notice that only krbtgt can open and read TGT data;
- Now the user can request a TGS ticket by sending a request called `TGS-REQ`" where it sends its TGT encrypted with the krbtgt hash to the DC/KDC and requests a ticket which will be used to access a specific service, (in our case the application server mentioned) this is the TGS ticket;
- The KDC is able to decrypt the TGT because it is the only component having access to the hashes. If the request is valid it provides the requester with a TGS encrypted using the target service’s account NTLM hash. This type of response is called `TGS-REP`;
- The user connects to the server hosting the service on the appropriate port and presents the TGS ticket. This request is called `AP-REQ`. The service will decide if the requesting user has privileges/rights to access the requested resource;
- (Optional) The application server may perform an optional mutual authentication with the client through a message called `AP-REP`;

- (Optional) In the Microsoft implementation of Kerberos, the application server may perform additional validation steps to check for authorization related to that specific user, this is related to the PAC (Privileged Attribute Certificate) `VERIFY_PAC_REQUEST`.

There can be an additional validation step to Kerberos called "PAC" (Privileged Attribute Certificate). Moreover, an additional Kerberos message is the `ERROR` message type which can be used to communicate error conditions.

From a security standpoint the additional details must also be taken into account:

- NTLM password hash for Kerberos uses RC4 encryption;
- TGT can be used as a Logon Ticket when authenticating to DC on a domain;
- DC validates user accounts only when the TGT is greater than 20 minutes;
- Service Ticket (TGS) PAC validation is optional and rarely used.

Figure 6.3 summarizes visually the main steps involved with the Kerberos authentication process.

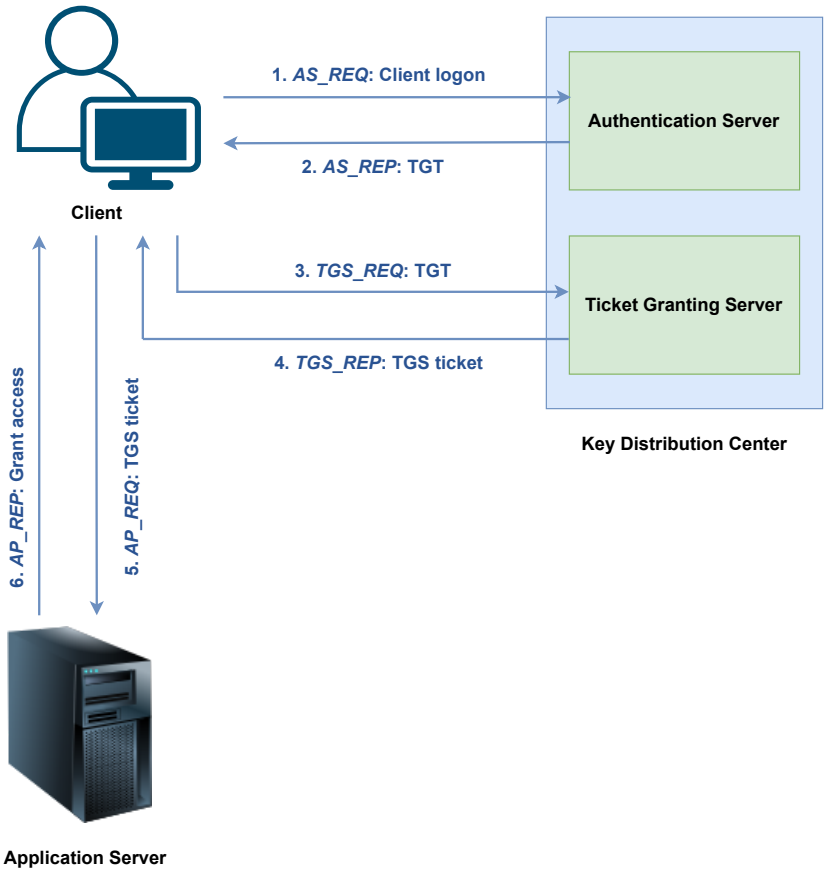


FIGURE 6.3: Kerberos authentication process

Note that, it is useful to distinguish between AS "Authentication Server", that is part of the KDC, hosted on the domain controller and AP that stands for "Application" and corresponds to the accessed service.

Active Directory has default settings that can be changed by system administrators, these settings provide an idea on common values for the validity of tickets. In particular:

- Enforce user logon restrictions: Enabled;
- Maximum lifetime for service ticket: 600 minutes (10 hours);
- Maximum lifetime for user ticket: 600 minutes (10 hours);
- Maximum lifetime for user ticket renewal : 7 days;
- Maximum tolerance for computer clock synchronization: 5 minutes.

Let us remark that Kerberos represents the core technology for the security within Microsoft Active Directory, hence it is frequently abused by attackers. Strong password requirements and password audits are necessary to ensure the security of environments relying on this complex protocol.

NetBIOS

NetBIOS (Network Basic Input/Output System) is a networking industry standard developed by Sytec in 1983 as an API for network communication on IBM networks. This API is described in RFC 1001 [140] and RFC 1002 [141]. NetBIOS allows applications on different computers to communicate over a Local Area Network. NetBIOS services were not originally designed to specifically run over TCP/IP networks. In fact, NetBIOS used to run over IEEE 802.2 networks, using NetBIOS Frames protocol (NBF) and on IPX/SPX networks, using NetBIOS over IPX/SPX protocol (NBX). In modern networks, NetBIOS runs over TCP/IP using the NetBIOS over TCP/IP protocol (NBT); consequently, every device on a NetBIOS enabled network a NetBIOS name. NetBIOS over TCP/IP was implemented to allow Windows 2000 and Windows XP computers to communicate with devices and share resources on the network running Windows operating systems.

A common misunderstanding is to associate a single protocol and port to NetBIOS, in reality, this protocol should be thought as a set of services running for different purposes and on different ports. More precisely, NetBIOS provides three different types of services:

- NetBIOS Name Service (NetBIOS-NS): enabling names registration and resolution;
- NetBIOS Session Service (NetBIOS-SSN): enabling reliable connection-oriented communications;
- NetBIOS Datagram Distribution Service (NetBIOS-DGM): enabling connectionless communications.

In what follows, a brief description of these three services is provided. Table 6.1 provides a summary of the services provided by NetBIOS and relative ports used.

NetBIOS Protocol	Purpose	Used Ports
NetBIOS-NS	Name Resolution	TCP 137; UDP 137
NetBIOS-SSN	Connection-Oriented Communication	TCP 139
NetBIOS-DGM	Connection-Less Communication	UDP 138

TABLE 6.1: Summary of NetBIOS services and corresponding TCP/IP ports

NetBIOS-NS: Name Service

NetBIOS Name Service is a network service for name resolution whose purpose is very similar to that of DNS, that is, translating human-readable names into IPv4 addresses (NetBIOS does not support name resolution for IPv6). NetBIOS-NS can use both TCP and UDP as transport protocol on port 137, even if UDP is used in most cases. Before the introduction of DNS, Active Directory relied mainly upon NetBIOS-NS and Windows Internet Name Service (WINS) for name resolution. Even if NetBIOS-NS provides the same service offered by DNS, it has different important limitations for modern networks who contributed to its loss in popularity in favor of DNS. The most important limitations are, flat name structure (in opposite to the hierarchical structure of DNS) and limited length of hostnames, which cannot have more than 15 characters. However, unlike DNS, it supports the peer-to-peer name resolution between all the computers within the same subnetwork. Nowadays, NetBIOS-NS is used in Active Directory for name resolution as a fallback solution whenever the DNS service is not available or is not able to resolve a specific hostname. For this reason, every domain controller is characterized by a NetBIOS name that is setup in addition to the DNS name.

The primitives provided by NetBIOS-NS are:

- Register a new NetBIOS name or a NetBIOS group name;
- Delete a NetBIOS name or a NetBIOS group name;
- Search for a NetBIOS name on the network.

Note that the NetBIOS-NS protocol on Windows is only used as a fallback protocol when DNS and LLMNR resolution fail.

The name resolution logic for NetBIOS follows a simple algorithm. If a Windows network node requires NetBIOS name resolution, it will first check its local NetBIOS computer name. It then looks at its local NetBIOS name cache for already acquired remote NetBIOS names. If no entry is found, the node forwards its NetBIOS query to the primary WINS server configured in the IPv4 properties of its network adapter. If the primary WINS server does not respond, it will query any other WINS servers it is configured for, if they exist. Only if no WINS server responds will the node send a broadcast NetBIOS query to the local subnet. Finally, if these attempts have been

unsuccessful, the node will check its LMHOSTS file and then its HOST file for the mapping. The flowchart representing the name resolution algorithm within the NetBIOS protocols is shown in Figure 6.4.

NetBIOS name service is particularly interesting from an attacker point of view since it is a commonly abused protocol to perform cache poisoning and MiTM attacks on a LAN network.

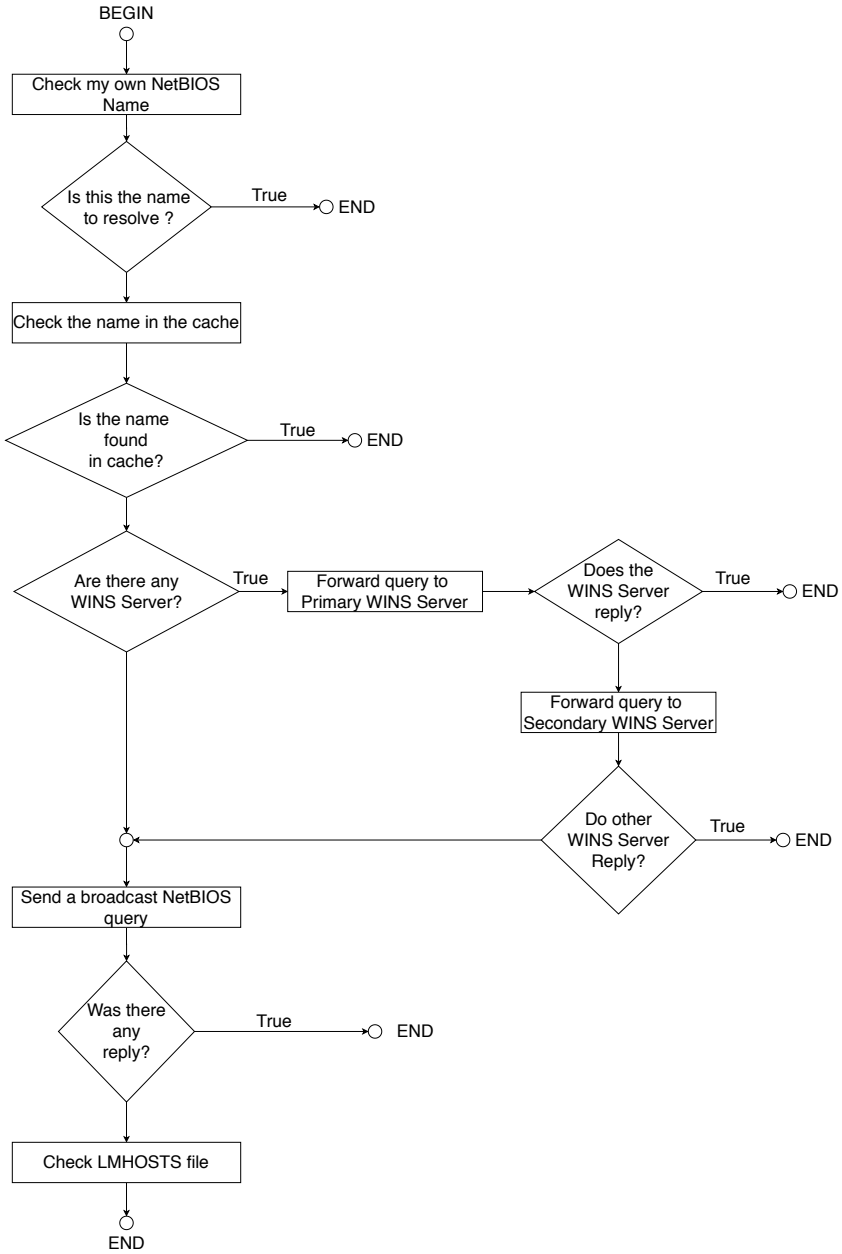


FIGURE 6.4: Flowchart describing the NetBIOS Name Resolution algorithm used in Microsoft Windows

NetBIOS-SSN: Session Service

NetBIOS Session Service was born as connection-oriented protocol analogous to the TCP protocol and nowadays runs on TCP port 139 by default.

This service allows two computers to establish a connection and transfer large amounts of data reliably. Furthermore, it enables authentication across a domain or a Windows Workgroup. NetBIOS-SSN provides features for error detection, data integrity checks and recovery.

The primitives offered by NetBIOS-SSN can be summarized as follows:

- Open a session to a remote NetBIOS name;
- Listen for attempts to open a session to a NetBIOS name;
- Send a packet to another end-system requiring an acknowledgement;
- Send a packet to another end-system without requiring an acknowledgement;
- Wait for a packet to arrive from the sender;
- Close a session.

The basic NetBIOS-SSN conversation in TCP/IP networks can be like this:

- Two NetBIOS enabled computers establish a TCP connection;
- Originator sends NetBIOS session request message (type=0x81) (this includes information about the caller and callee, like their NetBIOS names);
- Responder sends response (e.g., type=0x82 if session is accepted). Note that at this point of the communication the responder can also deny or redirect communication;
- Messages are exchanged. These are of type "session message" (i.e., type=0x00). The session message is 4 bytes NetBIOS specific information followed by the payload. SMB is typically carried in this payload. The first couple of bytes of the payload are known as SMB magic bytes. Note that, the NetBIOS message header is used to specify the length of the payload;
- NetBIOS can also take advantage of "Keepalive" messages to , whose type is "0x85";

NetBIOS-DGM: Datagram Distribution Service

NetBIOS Datagram Distribution Service provides a method for handling connectionless communication between computers. In the case of NBT, NetBIOS-DGM uses UDP on port 138.

The primitives provided by NetBIOS-DGM are:

- Send a datagram to a NetBIOS name;
- Send a datagram to all the NetBIOS names of the network (broadcasting);

- Wait for a packet to arrive from a sender;
- Wait for a packet that arrives from a sender that executed a Send Broadcast Datagram Operation.

NetBIOS: Security threats

NetBIOS services are commonly abused by attackers for different purposes. In fact, NetBIOS services can be abused both to perform enumeration against the target network or to remotely execute commands on a target machine.

The NetBIOS Name Service can for example be useful for an attacker in the enumeration phase, where it could be abused to perform:

- Enumeration of information regarding users, security policies and domains;
- Brute-forcing of user passwords.

On the other hand, the attacks that are typically performed against the NetBIOS datagram service are related to the enumeration phase, more precisely an attacker could gain:

- NetBIOS hostnames on a network;
- Accessible network interface MAC addresses;
- Authenticated users that are currently using a system;
- Domain name of which a specific system is a member.

In addition, if the attacker can authenticate into a NetBIOS session with appropriate privileges, the NetBIOS session service can be abused to perform the following operations:

- Access the system registry and edit registry keys;
- Run arbitrary command on the target host;
- Access the SAM password database;
- Transfer files.

Although as a security recommendation NetBIOS should be disabled, in plenty of Active Directory this is still enabled and abused by attackers.

RPC

Remote Procedure Call (RPC), is a protocol defined specified in RFC 1057 [142] and RFC 5531 [143]. This protocol uses the client-server model in order to allow one program to request service from a program on another computer without having to understand the details of that computer's network. Microsoft implements a custom version of RPC, called Microsoft RPC (MSRPC) [144], which is based on the DCE/RPC protocol. MSRPC is used

in Windows for implementing client-server functionalities; for example the Windows Server domains protocols completely relies on Remote Procedure Call. Some examples of services that use RPC are Outlook, Microsoft Exchange, and the Messenger Service. MSRPC is used also for replication in an Active Directory environment. MSRPC was originally derived from open source software but after additional development for Windows the implementation become closed source and copyrighted by Microsoft. Depending on the host configuration, the RPC endpoint mapper can be accessed through TCP and UDP port 135, via SMB with a null or authenticated session (TCP 139 and 445), and as a web service listening on TCP port 593. RPC provides a well-defined interface and a simple call syntax (high-level language) for communication between processes on the same machine (inter-process communication) or on remote machines. In RPC client-server model the device that requires a service is the client and the device that provides the service is the server.

A remote procedure call can be subdivided into six steps:

1. The client call the client stub through a local procedure call;
2. The client stub packs the procedure parameters into a message. The parameters are passed only by values;
3. RPC Runtime manages the transmission of the message between the client and the server over the network;
4. The message is received and unpacked by the server stub. After that the procedure is extracted, it is executed;
5. The server packs the result message using the stub and then it sends back the message to the client using TPC/IP as transport layer;
6. The client RPC Runtime pass the message to the stub which unpacks the message returning the parameters to the client.

In addition, remote procedure calls can be subdivided into three types:

- **CallBack RPC**: involved processes communicates each other according to a P2P paradigm; thus a each process can be both client and server;
- **Broadcast RPC**: the client's request is broadcast on the network and then it is processed by all the server that have the method for processing that particular type of request. Broadcast RPC allows to declare broadcast ports and help to reduce the network overhead;
- **Batch-mode RPC**: it allows the client to store separate RPC requests into a client-side transmission buffer. Once all the requests have been collected, they are sent as a single batch to the server The batch-mode helps to reduce the network overhead, but it requires a reliable transmission protocol as TCP and it is efficient only for the applications that require a low call rate.

Figure 6.5 presents the main steps involved with the RPC mechanism.

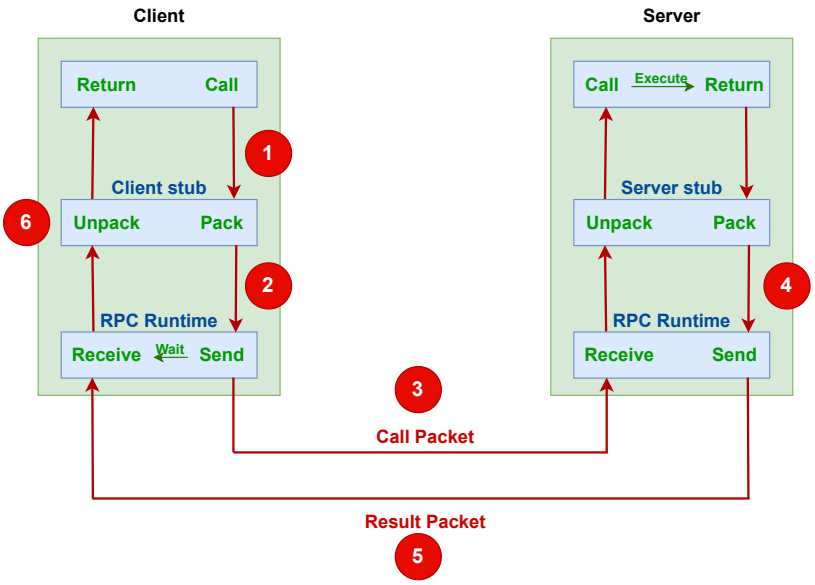


FIGURE 6.5: Remote Procedure Call mechanisms

Note that, RPC represents a fundamental building block for Windows, although many of its features are deprecated. Active Directory relies for many tasks on remote procedure calls, e.g., for replication tasks.

RPC: Security threats

RPC is a protocol widely used by developers since it is easy to use and provides advanced features for remote control of machines, but unfortunately it is also very popular with hackers. It is so powerful that it represents a security threat. For this reason, many organizations have restricted the functionalities offered by this protocol on their machines, sometimes reducing the productivity. In fact, RPC is a very chatty protocol that provides a lot of information about the services running on a server.

Exploiting the Microsoft Remote Call Procedure, an attacker can attempt to:

- Enumerate system information, such as IP addresses or network interfaces;
- Enumerate user details via the SAMR and LSARPC interfaces;
- Run commands through the Task Scheduler interface;
- Run arbitrary code and carry out Dos attacks;
- Brute-force the password of the users belonging to the Administrators group.

In 2003, when the Blaster worm was discovered, organizations learned that RPC could be widely exploited by malicious people. Blaster took advantage of a vulnerability in the Windows Distributed Component Object Model (DCOM) Remote Procedure Call (RPC) interface for taking control of a remote server, using an exposed RPC port.

Nowadays, most RPC traffic is blocked on the Internet, but attacks that exploits RPC are very common on the organizations' LANs. In fact, there are malware and viruses created specifically to conduct attacks on local networks exploiting the RPC protocol.

SMB

SMB (Server Message Block) is a client-server communication protocol for sharing files, printers and serial ports between computers on the network. Furthermore, it offers an authenticated inter-process communication mechanism. SMB is mainly used by the computers running Windows, for example, in the Active Directory environment.

When a client wants to start a new connection with the server, it connect to the server using NetBIOS over TCP/IP RFC 1001 [140], RFC 1002 [141], IPX/SPX or NetBEUI. After that a connection is established, the client sends specific commands to the server for accessing shares and files and, eventually, edit them. SMB with NetBEUI is supported by Windows 95, Windows 98, Windows NT and Windows 2000/Me. Starting from Windows XP, SMB/NetBEUI is no longer supported. In the past, SMB was often used with NetBIOS over TCP/IP (NBT) over UDP on port 137 and 138, exploiting the NetBIOS Name Service and the NetBIOS Datagram Distribution Service, or over TCP on port 137 and 139, exploiting the NetBIOS Name Service and the NetBIOS Session Service. NBT with SMB is supported by Windows 95, Windows 98, Windows NT, Windows 2000/Me, Windows XP and Windows Server 2003. Since Windows Vista and Windows Server 2008, SMB/NTB is used only for backward compatibility. In fact, Windows 2000 and later operating systems use SMB directly over TCP on port 445.

Let us remark that there are various SMB shares that are exposed by default The most important ones are:

- Default administrative shares, `C$` and `ADMIN$`;
- The inter-process communication share, `IPC$`;
- Domain controller shares, `NETLOGON` `SYSVOL`);
- Shared printer and fax shares, `PRINT$` and `FAX$`.

Note that the "$" symbol at the end of the share name is used to denote a default and hidden share. SMB protocol has two security levels:

- Share level: protection is applied at the share level on a server. Clients can connect to an SMB share and access to the files contained in it, only if it knows the password;

- User level: protection is applied to individual files on the base of ACLs.

These protections act as authorization mechanisms on resources but do not represent mitigations against the compromise of accounts. In fact, an attacker impersonating a user could access the resources associated with the compromised account.

Microsoft SMB Implementations

Microsoft has implemented three major versions of SMB, namely:

- CIFS (also known as SMBv1);
- SMBv2;
- SMBv3.

In its first version, SMB ran over NetBIOS/NetBEUI API, but, starting from Windows 2000, it runs over TCP on port 445, exploiting a feature known as Direct Host SMB. In 1996, Microsoft launched an initiative for renaming SMB to CIFS (Common Internet File System). Furthermore, Microsoft implemented new features, such as support for larger file sizes, hard links and symbolic links. SMB 1.x was implemented in Windows 2000 and it was deprecated in June 2013, starting from Windows Server 2012 R2. Moreover, SMB 1.x is not installed by default in the later versions of Windows, such as Windows Server 2016 and Windows 10 Fall Creators Update (1709). In 2006, Microsoft implemented SMB 2.0 in Windows Vista. SMB 2.0 reduces the set of SMB commands to just nineteen. It has a new mechanism for pipelining, it improves the performance over high latency links and introduces the possibility of compounding multiple actions into a single request in order to reduce the number of round-trips that the client has to do to the server. SMB2 implements a mechanism for allowing the SMB connection to survive to brief network outages, without incurring in the negotiation of a new session. Moreover, SMB2 improves the caching of file properties, the scalability, increasing the number of users, shares and open files per server, and performance in large file transfers using 32 or 64-bit wide storage fields, and 128 bits in the case of file-handles. In this way, the constraints on block sizes, imposed by SMB 1.x that uses 16-bit data sizes, have been overcome. SMB 2.0 is used by default in Windows Vista Service Pack 1 and Windows Server 2008. In Windows 7 and Windows Server 2008 R2, Microsoft implements SMB 2.1. This minor update, enhances performance exploiting a new opportunistic locking mechanism. SMB 3.0 was implemented for the first time in Windows 2008 and Windows Server 2012. It brought several new features for improving performance and functionality in data centers, such as SMB Multichannel, for supporting multiple connection for SMB session, SMB over Direct Memory Access (RDMA), and SMB Transparent Failover. Furthermore, in order to increase security, Microsoft implemented a new signing algorithm based on AES and end-to-end encryption. In Windows 8.1 and Windows Server 2012 R2, Microsoft updated SMB to the version 3.0.2, adding the possibility of disabling SMB for improving security. In Windows 10 and Windows

Server 2016/2019, SMB 3.1.1 is used by default. SMB 3.1.1 adds the support for AES-128 GCM encryption, implements pre-authenticating integrity check using SHA-512 hash and makes security negotiation mandatory for connection using SMB 2.0 or higher.

SMB: Security threats

Using an anonymous SMB session, it is possible to access the `IPC$` share and query the services exposed via named pipes. In this way it is possible to retrieve details on the parent domain and on the operating system, information on the local users and groups, a list of available SMB shares and details on the security policies.

Some of the most dangerous ransomware and trojan malware exploit typical vulnerabilities of the SMB protocol. In fact, in 2018, MalwareBytes, exploiting its antivirus telemetry services, conducted a research which discovered that within 30 days there were 5315 installations of Emotet and 6222 installations of TrickBot (two trojans exploiting SMB vulnerabilities) in the business networks analyzed. [1]

In March 2017, Microsoft patched these vulnerabilities, but many organizations or private users have not installed do not upgrade regularly their device with the last updates available.

Nowadays, there are three well-known vulnerabilities that exploit SMB vulnerabilities:

- EternalBlue, exploited by Emotet and WannaCry;
- EternalRomance, exploited by TrickBot;
- EternalChampion.

EternalBlue relies on a bug in the process of converting File Extended Attributes (FEA) from OS2 structure to NT structure by the Windows SMB implementation. This bug can lead to a buffer overflow in the non-paged kernel pool. In this way, attackers can control the content of forbidden memory locations. In the specific case of EternalBlue, attackers can take the control of a heap that has execution permissions, and, exploiting a Remote Code Execution vulnerability, they can take the control of the target machine executing remote commands

EternalRomance is a Remote Code Execution attack which exploits CVE-2017-0145 against SMB 1.x. EternalRomance is based on a type confusion vulnerability. This type of vulnerability is a programming flaw that happens when a piece of code is executed without verifying the type of object that is passed to it. In this way, an attacker can feed function pointers and data into the wrong piece of code in order to attempt to execute remote commands.

[1] https://blog.malwarebytes.com/101/2018/12/how-threat-actors-are-using-smb-vulnerabilities/

EternalChampion exploits a race condition in how SMB 1.x handles transactions. Exploiting this bug, an attacker can change the sequence or the timing of a series of events in order to try to get a Remote Code Execution and make information leak.

RDP

RDP (Remote Desktop Protocol) [145] is a proprietary secure network protocol developed by Microsoft, which allows a user to connect to another computer exploiting a graphical interface. RDP is an extension of the T-120 family of protocol standards, and, by default, it runs on port 3389 and uses both TCP and UDP as communication protocol. RDP provides 64000 separate channels for data transmission, even if current transmission activities use a single channel for mouse, keyboard and presentation data. The screen is transmitted as a bitmap from the server to the client and the client transmits mouse and keyboard interactions to the server.

RDP was originally designed to run over IPX and NetBIOS protocols; however, nowadays, RDP runs only over TCP/IP. In the future, RDP may support communication protocols other than those offered by TCP/IP because it was designed to be completely independent of its transport stack.

RDP has only four primitives:

- Connection request;
- Connection confirmation;
- Handle data transmission;
- Disconnection request.

RDP relies on a security layer which includes encryption and signature services. This security layer prevents malicious users from intercepting, monitoring and manipulating data exchanged during an RDP session. The algorithm used for encrypting data is the RC4 by RSA Inc. Instead, for preventing data manipulation, RDP leans on a signature mechanism that consists of a combination of the MD4 and SHA-1 algorithms.

When a client wants to start an RDP connection with a terminal server, it has to send a set of parameter and information about it, that the server will use in order to improve the user experience. For example, the client needs to send information to the server about the RDP protocol version, the operating system, the data compression supported, the desktop size, the preferred color depth, the set of local characters supported and many other specifications.

The main problem that had to be faced in the development of RDP was related to the huge amount of data transmitted between client and server. Thus, for reducing the data transmitted on the network, two mechanism

were implemented: data compression and caching. Data compression consists in reducing graphical element to the minimum necessary, removing animations and decorations, and reducing the number of colors. Caching, instead, consists in temporary store on the client frequently used images fragments that can be displayed again. There are also special network components that can store RDP, providing a global caching space on a LAN segment.

The Windows versions that have installed a Remote Desktop Connection client are Windows NT 4.0 Terminal Server Edition, Windows 2000 Server, Windows XP Professional, Windows Fundamentals for Legacy PCs, Windows Home Server, Windows Server 2003 and latest, Windows Vista, 7, 8.x, 10 (excluded Home editions).

RDP: Security threats

The main threats related to the Remote Desktop Protocol are:

- Brute-forcing attack;
- Man-In-The-Middle attack;
- Cryptographic attack;
- Pass the hash attacks;
- DoS attack.

In a brute-forcing attack, malicious users can scan the network to searching for Windows devices that have port 3389 open. Then, they can run automatic tools that try to guess the login credentials starting from lists of common usernames and passwords. A study conducted by Microsoft states that in 90% of cases brute-forcing attacks last from a few days to a week. Only in 5% of cases the brute-force attacks last for two weeks or more. According to Microsoft, only the 0.8% of the attacked computers analyzed by the Windows telemetry were compromised during an RDP attack. Brute-forcing attacks last days and not hours in order not to trigger the Intrusion Detection system (IDS).

RDP offers data encryption, but not provide authentication for checking the identity of the Terminal Server. Thus, malicious users can exploit the lack of identity verification in order to intercept the network traffic exchanged between the client and the server. Attackers can perform DNS spoofing or ARP spoofing in order to redirect data transmitted from the RDP client to the attacker's computer before transmitting them to the Terminal Server.

By default, RDP encrypts data exchanged between the client and the server using the maximum key strength supported by the client. Thus, if an old version of RDP client with week encryption settings is used, it will be possible to decrypt the connection in a reasonable time-frame.

RDP sessions are susceptible to in-memory credential harvesting; thus an attacker can retrieve them and carry out a pass the hash attack. Exploiting a pass the hash attack, it is possible to authenticate to a service without

stealing or cracking the credentials of a user. Thus, in this way, an attacker can authenticate to the Terminal Server using the NTLM hash instead of the plaintext account's password.

Terminal Servers which have not configured Network Level Authentication (NLA) are susceptible to Denial of Service attacks (DoS). NLA is a feature of RDP that compels a user to authenticate to the service before starting an RDP connection with the server. This feature is very important because the creation of a new session is an expensive task from the point of view of the resources used. Thus, enabling NLA, it is possible to reduce the susceptibility to DoS attacks.

Net-NTLM

Net-NTLM (Net-NT LAN Manager) [146] is a family of security protocols developed by Microsoft with the aim of providing users with authentication, integrity and confidentiality.

The Net-NTLM suite includes LAN Manager (versions 1 and 2), Net-NTLM v.1 and Net-NTLM v.2. Net-NTLM is an encrypted challenge/response authentication protocol that has the task of proving to the domain controller that the user knows the account's password. An interactive Net-NTLM authentication procedure consists of three steps:

1. The client establishes a connection with the server and sends a packet known as `NEGOTIATE_MESSAGE` in which it specifies its capabilities;

2. The server replies with a `CHALLENGE_MESSAGE` which is necessary for establishing the identity of the client. The challenge consists of a 16-bit random number, called nonce;

3. The client solves the challenge encrypting the nonce with the hash of the account's password. Then, it sends the result to the server with an `AUTHENTICATE_MESSAGE`.

Net-NTLM can run on both TCP and UDP and the port used is that of the service which relies on Net-NTLM as the authentication protocol. Although Kerberos is the preferred authentication method in Active Directory since Windows 2000, Net-NTLM authentication is still used for Windows authentication with systems configured as members of a workgroup or for logon authentication on non-domain controllers. Furthermore Net-NTLM is used for authenticating non-Windows machines and as secondary authentication method for the SMB protocol. Microsoft recommends not using Net-NTLM in applications because, as specified in Microsoft documentation, "NTLM does not support any recent cryptographic methods, such as AES or SHA-256. It uses cyclic redundancy check (CRC) or message digest algorithms (RFC1321 [147]) for integrity, and it uses RC4 for encryption. Deriving a key from a password is as specified in (RFC1320 [148]). Therefore, applications are generally advised not to use NTLM" [149]. Nowadays, Net-NTLM authentication is supported in the latest version of Windows only for compatibility with older systems.

Net-NTLM: Security threats

Net-NTLM authentication does not support the typical security features of Kerberos v.5. For example, Net-NTLM does not support mutual authentication, multi-factor authentication and a modern encryption method. Attackers often try to compromise the target Active Directory infrastructure exploiting the vulnerabilities of the Net-NTLM authentication method. The main type of attack to which Net-NTLM is susceptible are due to the lack of mutual authentication mechanism. These attacks are the NTLM relay attack and Person-In-The-Middle attack. As shown in Figure 6.6, when a target user wants to connect to a server via Net-NTLM, an attacker could hijack the connection and relay the user's credentials. In this way, the attacker could authenticate to the server using the challenge encrypted by the user.

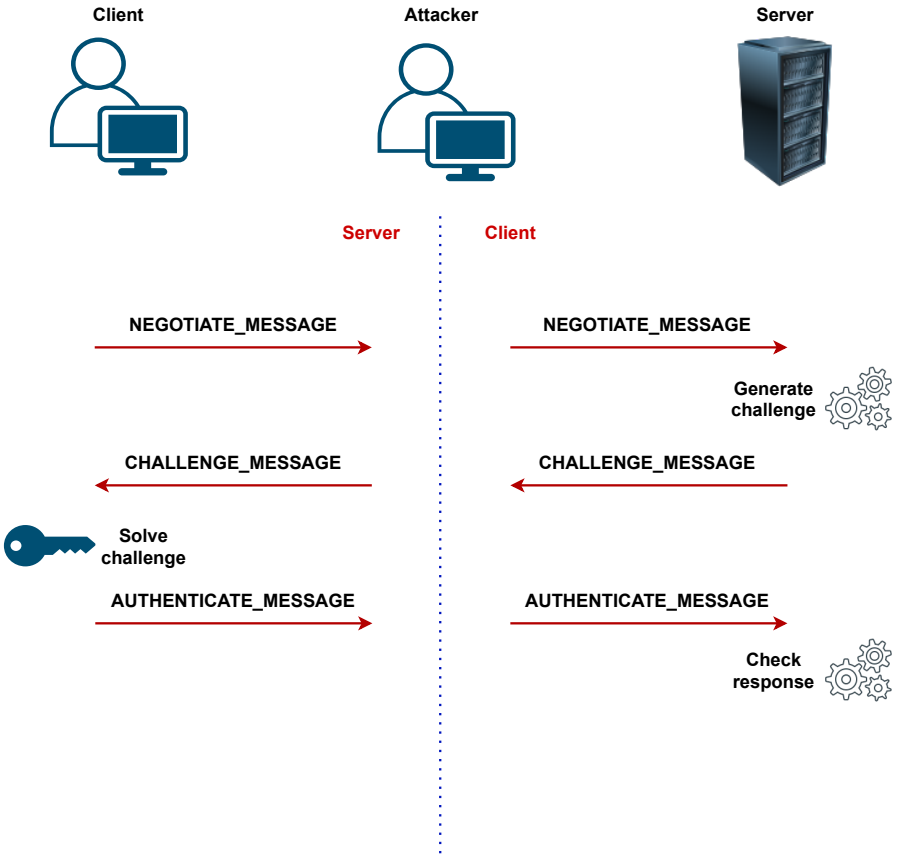


FIGURE 6.6: Net-NTLM Relay Attack

Note that this type of attack represents one of the main techniques for performing lateral movement by attackers who already have a foothold within a network.

LLMNR

LLMNR (Link-Local Multicast Name Resolution), as specified in RFC 4795 [150], is a protocol over IPv4 and IPv6 that allows name resolution for hosts on the same local link when the DNS is not available.

In responding to queries, computers that use LLMNR protocol listens on UDP port 5355 on the following link-scope multicast addresses:

- IPv4: 224.0.0.252, MAC address: 01-00-5E-00-00-FC;
- IPv6: FF02:0:0:0:0:0:1:3, MAC address: MAC address 33-33-00-01-00-03.

The responders listen on TCP port 5355 on the unicast address that the host uses to respond to queries.

Typically, an LLMNR sender sends an LLMNR query to the link-scope multicast address, unless a unicast query is indicated. Then, a responder responds only if it is authoritative for the name contained into the query. The responder responses to a multicast query by sending a unicast UDP response to the sender. Finally the sender processes the response.

The Microsoft implementation of LLMNR, known as MS-LLMNRP, differs from the version described in RFC 4795 [150] in the area of transport. In fact, TCP support and EDNS0 (RFC 6891 [151], RFC 2671 [152]) support are optional.

LLMNR: Security threats

The main security threats is due to the lack of mutual authentication; in fact, an attacker can spoof an authoritative source for name resolution pretending to know the identity of the requested host. If the target user tries to query for a host that requires identification and/or authentication, he will send his username and the NTLMv2 hash to the attacker device. Thus, the attacker can collect the hashes transmitted over the network using a sniffer and then he can try to crack the hashes offline using brute-forcing techniques in order to retrieve the plaintext password. It is possible to crack the hash in a reasonable time since NTLMv2 relies on MD4 hash algorithm.

CIFS Browser Protocol

As explained in the Microsoft documentation, the Common Internet File System (CIFS) Browser Protocol is a protocol that operates on the top of SMB and it is used for discovering machines and the resources on the network.

The CIFS Browser Protocol, for example, is used by an application in order to identify print servers and files on a local subnet.

The CIFS Browser Protocol allows:

- A server or a set of servers to act as a browser server for retrieving information about the services available on the network;
- A set of servers that are making services available to access the browser server and advertise the services they provide. These server are called non-browser servers;
- A set of clients, called browser clients or workstations, to access the information provided by the browser server and query details about a particular service.

The CIFS Browser Protocol relies on NetBIOS name service for host name resolution and on the Remote Mailslot Protocol for delivering datagrams between the end-points.

The CIFS Browser Protocol manages groups of computer. The computers belonging to these groups can be members of a workgroup or of a domain. If computers are members of an Active Directory domain, the CIFS Browser Protocol allows for browsing across multiple subnets by exploiting the domain master browser server. It is usually the Primary Domain Controller (PDC) that covers the role of domain master browser server and queries all the local master browsers of the domain for getting a list of all domains and servers known in their subnets.

For each machine group, a single local browser master is selected. The selection is made through an election process. Local master browsers periodically query the domain master browser for getting network-wide information. There are also backup browsers that maintain a replica of the information stored on the local master browser. For this purpose, they periodically synchronize with the local master browser.

CIFS Browser Protocol: Security threats

In February 2011 a vulnerability was discovered affecting the CIFS Browser Protocol. The vulnerability can lead to a Denial of Service and, theoretically, it can also lead to a Remote Code Execution. All versions of Windows were vulnerable. In particular, it affects all the computers that have been configured to use the CIFS browser protocol and that are or could become master browser on the local network.

The vulnerability can be triggered by a malformed browser message sent to a master browser and it is due to an integer underflow where a 32-bit length value becomes -1. This vulnerability lead to a kernel buffer overrun causing a Denial of Service. Remote Code Execution is possible only if the corrupted memory is used before the RtlCopyMemory triggers a bug-check, and in a way that can be used to modify the executed code.

Web Service-Management Protocol

As specified in Microsoft documentation [153], Web Service-Management (WS-Management) is a DMTF (Distributed Management Task Force) open-standard protocol for the management of devices, servers, applications and web services. WS-Management offers a method for accessing and exchanging management information across the IT infrastructure. It is used on Windows machines as an alternative to ssh to control devices remotely.

It relies on SOAP (Simple Object Access Protocol) over HTTP, SOAP 1.2, HTTPS, WS-Addressing, WS-Enumeration, WS-Transfer and WS-Eventing, as described in DMTF specifications [154].

WS-Addressing and WS-Transfer are the standards that define XML schemas for web service messages. The messages refer to a resource using a resource URI. WS-Transfer defines a set of operations and values for managing remote resources; for example it defines the operations GET, PUT, CREATE

and DELETE. Furthermore WS-Management includes RENAME, PARTIAL GET and PARTIAL PUT in its set of operations

The XML messages are assembled by Windows Remote Management when the user execute a command using the Windows Remote Management (WinRM) command-line tool and then they are sent using the Sample Object Access Protocol over HTTP (port 80) or HTTPS (port 443). Starting from Windows 7, the default ports used by WinRM are 5985 for SOAP over HTTP and 5986 for SOAP over HTTPS. The user can send a WS-Management message only if his account is associated to the local administrator groups on the remote computer or he is member of the Administrators group.

Microsoft implemented WS-Management for the first time in WinRM 1.1 that was available for Windows XP, Windows Vista, Windows Server 2003 and Windows Server 2008. On October 2009, WinRM 2.0 was released introducing the possibility of invoking PowerShell commands on one or multiple remote machines at the same time. Since Windows 7 and Windows Server 2008 R2, the most updated version of WinRM is the 3.0.

6.3 Active Directory Security

In this section, the major threats related to Active Directory environments are described. In particular, access control abuses and Kerberos related attacks are discussed. Active Directory has always represented a huge attack surface for attackers, because of the complexity associated with these environments and the significant amount of features it provides to its users. Although there are many interesting attack vectors to be aware of, this section focuses on the two most prevalent areas of interest for attackers, that are, the Single Sign On infrastructure provided by Kerberos and the rich access control mechanisms offered by Active Directory to system administrators.

6.3.1 Kerberos Attacks

Kerberos represents the core of security for Active Directory environments, hence it is frequently targeted by attackers. Over the years, different important attacks have been developed relying on abuse of Kerberos functionalities. These attacks are commonly used in penetration testing activities and red teaming operations. In this section a summary of the main attacks is provided and for each attack both pre-conditions and post-conditions are detailed. More precisely, pre-conditions represent what are the prerequisites that an attacker must have in order to perform the attack. On the other hand, post-conditions represent the outcome of an attack.

The summary of the main attacks related to Kerberos in Active Directory are summarized in the Table 6.2.

Login Brute-force

In first place, due to the Kerberos nature of being an authentication protocol, it is possible to perform brute-force attacks against it. Moreover, performing

Attack	Type of Attack	Pre-Condition	Post-Condition
Login Bruteforce	Enumeration/ Online Brute-force	No domain access required (Optional) Valid usernames	A set of valid users A set of valid credentials
AS-REPRoasting	Offline Brute-force	A set of users with pre-authentication disabled	A set of hashes
Pass the Key	Lateral Movement	A set of NTLM hashes or passwords	A set of TGT tickets
Pass the Ticket	Lateral Movement	A set of TGT tickets;	Authentication with TGT
Unconstrained Delegation	Lateral Movement	Access to a computer with unconstrained delegation	A set of TGT tickets
Constrained Delegation	Lateral Movement	Access to a computer with constrained delegation	A set of TGS tickets
Kerberoasting	Offline Bruteforce	A set of valid user credentials	A set of hases
Golden Ticket	Persistence	Domain Admin access	Hash of krbtgt account
Silver Ticket	Persistence	NTLM hash of a service account	Access to a service with any user
Skeleton Key	Persistence	Domain Admin access	Access as any user with a single password

TABLE 6.2: Kerberos main attacks breakdown

a brute-force attack using Kerberos has many advantages over brute-forcing other authentication methods, these advantages can be summarized as follows:

- No domain account is needed to conduct the attack, just connectivity to the KDC (i.e., a domain controller);
- Kerberos pre-authentication errors are not logged in Active Directory with a normal Logon failure event (4625), but rather with specific logs to Kerberos pre-authentication failure (4771), making them more difficult to detect by System Administrators;
- The Kerberos authentication process provides information about the correctness of the username even if the provided password is wrong. This is very important from the perspective of an attacker because they can use this technique to collect valid usernames;
- In Kerberos brute-forcing it is also possible to discover user accounts without pre-authentication required, which can be useful to perform an ASREPRoasting attack.

Note that depending on the configuration set on the domain controller, by carrying out a brute-force attack it is also possible to block user accounts. Thus, this technique should be used carefully from an attacker perspective.

AS-REPRoasting

If we are able to find any accounts in a Windows domain that don't require Kerberos pre-authentication, we can easily request a Ticket for these accounts and try to crack the encrypted part of these Tickets offline, revealing the user's credentials.

Basically in the first step of a Kerberos communication the client performs an `AS-REQ` requests where he tries to identify and authenticate itself by sending the current timestamp encrypted with its NTLM hash. The Kerberos Server (Authentication Server) verifies if the user claims who it is (since he has the hashes of any user in the domain) and provides him/her with a TGT in the `AS-REP` response. Unfortunately, this kind of first authentication is skipped if the specific use has pre-authentication disabled `DONT_REQ_PREAUTH`. Hence a TGT for these users can be requested without any kind of validation. While the `AS-REP` ticket itself is encrypted with the service key (in this case

the krbtgt hash) the AS-REP "encrypted part" is signed with the client key, i.e. the key of the user we send an AS-REQ for. Hence we can attempt to offline brute-force the user key and retrieve its password.

Note that as a matter of fact the reason for enabling Kerberos pre-authentication is to prevent offline password brute-force.

Kerberoasting

Kerberoasting takes advantage of how service accounts leverage Kerberos authentication with Service Principal Names (SPNs). From an attacker point of view it is very important to find service accounts. Kerberoasting allows us to crack passwords for those accounts. By logging into an Active Directory domain as any authenticated user, we are able to request service tickets (TGS) for service accounts by specifying their SPN value. A KDC will provide as response the encrypted service ticket, (is encrypted using the NTLM hash of the service account that is associated with the specific SPN). We can then brute force these service tickets until successfully cracked, with no risk of detection or account lockouts.

This attack can be summarized as:

- Scan Active Directory for user accounts with SPN values set;
- Request service tickets from AD using the found SPN values;
- Extract service tickets to memory and save to a file;
- Brute-force attack those passwords offline until cracked.

Kerberoasting applies to service accounts, so it is important to use strong passwords for service accounts.

Pass The Key

Pass the Key attack, also known as Overpass The Hash attack, is referred to an attack where a user uses its NT hash to request a valid TGT ticket. The "over" in Overpass-The-Hash refers to taking the pass-the-hash technique one step further to acquire a valid Kerberos ticket.

Typically, with Pass-The-Hash you use a NT hash from a compromised user account to directly authenticate to remote services as that user, either by injecting into the memory of the current Windows user or by providing the hash directly to client applications which are able to accept NT hashes.

With Overpass-The-Hash we can leverage an NT hash twice to request a valid Kerberos TGT (or TGS) from the KDC on behalf of the compromised user. This technique is often used in combination with the Pass-The-Ticket attack, where a forged valid (not expired) TGT or TGS can be exported and re-injected for future use to bypass communication with the KDC.

Since this attack takes advantage of the NT hash to request Kerberos tickets, as an alternative to the common Pass-The-Hash over NTLM protocol, this technique becomes particularly useful in networks where the Net-NTLM protocol is disabled and only Kerberos is allowed as authentication protocol.

In order to perform this attack, an attacker has to gather the NT hash related to the password of a target domain user. Once the NT hash is obtained, a TGT can be requested for that account, hence typically an attacker is able to impersonate a domain user. By taking advantage of pass-the-key attacks an attacker is generally able to access any service or machine where the target user account has permissions.

Pass the Ticket

Pass-the-ticket is a credential theft technique that enables attackers to use stolen Kerberos tickets to authenticate to resources (e.g. file shares or other computers) as a user without compromising that user's password. This technique is often used by adversaries to perform lateral movement within the target network. This kind of attack is similar to Pass-the-Key, but instead of using user hashes to request tickets, the ticket itself is stolen and used to authenticate as its owner.

In general an attacker must have compromised a user account, a computer containing readable tickets or obtained tickets in some other ways. This attack is commonly used in combination with the Pass-The-Key technique.

Note that both TGT tickets and TGS tickets can be stolen and reused by adversaries. Without administrative privileges, an adversary can obtain the TGT (using "fake delegation") and all TGS tickets for the current user. Moreover, if an attacker has administrative privileges on a target machine, it can dump the LSASS process and obtain all TGTs and TGS tickets cached on the system.

Golden Tickets

Golden tickets attack are used for persistence purposes once we have conquered a domain admin account and are able to dump the hash of the krbtgt account. Basically this is possible if we are able to forge the TGS-REQ by using a valid TGT ticket. The scenario here is that once we have obtained the krbtgt hash (and to obtain this we must have conquered a domain controller) we can access any service with any account since we have the key which is able to give us any TGS for any user for any service. Basically having this key we can forge whatever request with any user account for any service.

We only need a valid TGT encrypted by the Kerberos krbtgt account. Since account validation is not done by the KDC until the TGT is older than 20 minutes we can also take advantage of pretending to be disabled/deleted accounts. This kind of ticket can be used to impersonate any account since we can write anything in our request, since it is enough to encrypt it with the krbtgt hash.

Notice that if we are able to get the hash of the krbtgt account on the domain controller, we can impersonate basically any user with any privileges on the domain. Moreover, note that password changes have no effect on this attack.

Silver Tickets

Silver tickets attacks are used for persistence for a specific service. Basically this is possible if we are able to forge the `AP-REQ` (step 5) by using a valid TGS ticket. Basically this happens when we have access to the NTLM hash of a service account, allowing us to forge valid TGS tickets as any user for that service.

Silver tickets have a reasonable persistence of 30 days for computer accounts. Generally we are very interested in the machine account, which is indeed used as a service account for many different services.

Comparison between Golden and Silver Tickets

Golden tickets allows us to impersonate any user and request any service, since by having the hash of the `krbtgt` account we can forge our own TGS with any detail we want. This is the holy grail of domain persistence.

Silver tickets allows us to impersonate any user on a specific service, since by having the hash of the service account we can forge our own TGS for that service with any user detail we want.

Skeleton Key

Skeleton key is another persistence technique where it is possible to patch the `lsass.exe` process of a domain controller so that it allows access as any user with a single password. This is a very effective backdoor which was originally found in a malware named "Skeleton key malware", anyway this technique is not used often by penetration testers since, the backdoor does not persist across Domain Controller reboots. Indeed, the less the domain controllers are rebooted in a company the more effective this technique becomes.

When this attack is used anyway, attackers try to install this skeleton key on each Domain Controller to increase its effectiveness, so that when a domain controller goes down, the skeleton key can still be used thanks to other domain controllers. Skeleton key allows access also to other machines in the domain.

Note that in order to patch the `lsass.exe` process on a domain controller an attacker must have domain administrative rights, hence a full compromise of the domain. For this reason, when a domain is fully compromised attackers prefer more stable persistence techniques such as Golden Keys or Silver Keys.

6.3.2 Access Control List Abuses

During penetration testing activities and red team engagements, there are a number of Active Directory misconfigurations that are commonly found which depend on incorrect access control list settings.

An ACL is a set of rules defining which entities have which permissions on a specific AD object. These objects are generally user accounts, groups,

computers, the domain itself and other entities. The ACL can be configured on an individual object such as a user account, but can also be configured on an Organizational Unit (OU). The main advantage of configuring the ACL on an OU is that it allows an ACL to be applied to all objects within the OU. The ACL of the Organizational Unit (OU) wherein the objects reside, contains an Access Control Entry (ACE) that defines the identity and the corresponding permissions that are applied on the OU and descending objects. The identity that is specified in the ACE does not necessarily need to be the user account itself; it is a common practice to apply permissions to AD security groups. By adding the user account as a member of this security group, the user account is granted the permissions that are configured within the ACE, because the user is a member of that security group.

- `AllExtendedRights`: these are extended rights granted on objects which allow a principal to read privileged attributes and perform "administrative" actions, for example adding principals to a group or change a target user's password are both examples of Active Directory extended rights. If an attacker compromises a user or a group that presents an AllExtendedRights ACL toward an important AD object, that object is compromised;

- `GetChangesAll`: when a user has this right in relation to a domain controller, it is able to replicate objects from the domain, hence obtaining sensitive information. The abuse of this relationship is also known as "DCSync attack";

- `AddMember`: when a user has the `AddMember` right with respect to a group, it has the ability to add arbitrary security principals (including itself) to the target group. The advantage of adding a user to a group, is that the newly added user will have the same privileges of the target group;

- `Self`, or `Self-Membership`: when a user has the `Self` right with respect to a group, it has the ability to add itself to that target group. The advantage for an attacker of being able to add itself to a group is that it will acquire all the privileges valid for that target group;

- `ForceChangePassword`: when a user has the right to change the password of another user without having to know its current credentials. The advantage of being able to change a password, is that it allows an attacker to impersonate a target user. Note that these kind of attacks are easily detected, anyway attackers may consider useful to perform these attacks targeting users with a low login frequency and who are not logged in;

- `GenericAll`: a user has a `GenericAll` right to an object when it has full control over that object. This privilege allows the trustee to manipulate the target object in any way;

- `GenericWrite`: a user has a `GenericWrite` right to an object when it can write to any non-protected attribute on the target object, including

Members of a group or Service Principal Names for a user. This grants an attacker the ability to add itself to a group and obtaining the privileges that come with that group or perform targeted Kerberoast attacks by adding to that user the service principal name of a service we want to attack;

- `Owns`: a user has a `Owns` right with respect to an object when it can modify the object security descriptors independently from the object discretionary access control list;
- `WriteOwner`: a user has a `WriteOwner` right with respect to an object when it can change the owner of an object, hence acquiring the same privileges of an `Owns` relationship. Basically an attacker may modify the security descriptors of an object independently from the object discretionary access control list;
- `ReadLAPSPassword`: a user has a `ReadLAPSPassword` right with respect to a computer, when it can read the password related to the local administrator of that computer. The local administrator password for a computer managed by LAPS is stored in the confidential LDAP attribute `ms-mcs-AdmPwd`.

Note that Active Directory objects such as users and groups are securable objects and ACE/DACL define who can read or modify those objects. Nonetheless, misconfigurations within access control mechanisms in enterprise environments is very common and this makes access control one of the favorite targets for attackers.

In fact, access control misconfigurations in these context represent a common attack vector. In conclusion, modeling ACLs and access control mechanisms as graphs, as described in this thesis work, is particularly beneficial since graphs highlight non-obvious chained misconfigurations that could lead to a network compromise.

Chapter 7

AI-based Framework for Security Assessment

This chapter presents the methodological approach proposed in this thesis work for AI-assisted security assessment of complex technological environments. More precisely, the chapter introduces the workflow for the offline detection of vulnerable networks and discusses the proposed classification approach aimed at evaluating the security of a complex environment and determining whether it is vulnerable.

7.1 Methodological Approach

The evaluation of computer network security is a complex task. Despite the availability of some automatic tools customized to specific areas of assessment, manual procedures are still highly valuable because of the complexity of the environments under examination. This complexity is due to the number of computers, users, organizational units, technologies and protocols used within these environment. In general, manual procedures are coupled with automatic scanners that determine the presence of vulnerabilities by using heuristics. Automatic tools save a significant amount of time in penetration testing activities, nonetheless they have two main disadvantages. The number of false positives they produce can be large. Moreover, automation based on heuristics is difficult to develop when the number of heuristics is significant. For this purpose in the last years the attention for the development of AI-based network assessment tools has increased. Nevertheless, there is a lack of research and toolsets to tackle security assessment in complex enterprise networks. For this purpose, in this thesis work a methodological approach for the systematic evaluation of the security status of complex environments is proposed. This approach provides users with a framework to automatically evaluate their networks by using a machine learning based classification system.

This novel framework combines graph theory and machine learning techniques in order to extract useful insights from graph models and transform these insights into features that are fed in a classifier.

Note that, in this context, although there is a growing interest in applying machine learning to the computer security field, the scientific literature

does not include works that consider AI-assisted network evaluation for enterprise environments. For this purpose, there is an interest (and need) to incorporate artificial intelligence techniques within network security assessment tools and penetration testing methodologies.

The overall architecture of the proposed framework is shown in Figure 7.1. In this architecture we can identify five main steps dealing with: data acquisition, definition of a knowledge base used to build network graphs and network classification where a classifier determines whether the network is vulnerable or safe according to the features extracted from the graphs. In what follows details about these steps are given.

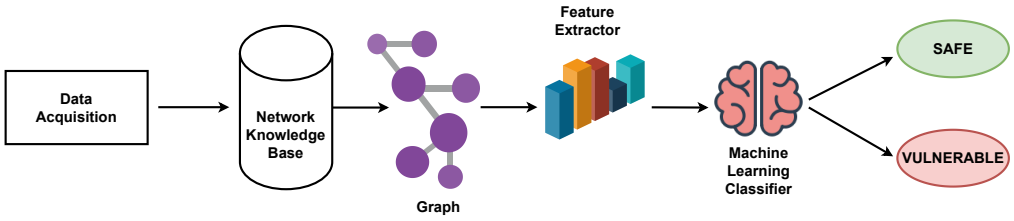


FIGURE 7.1: Architecture of the proposed framework

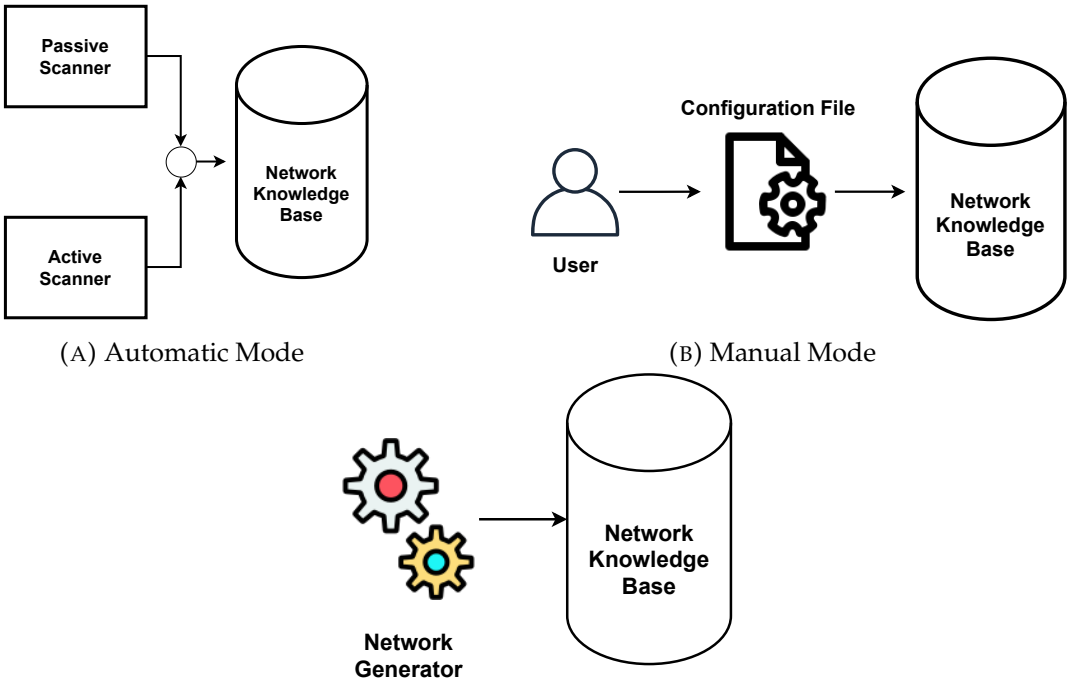
7.2 Data Acquisition Phase

The first step of the methodological approach is related to the acquisition of data about the target being analyzed. The output of this phase is data organized in a structured knowledge base to be further analyzed in subsequent steps. The input data refers to:

- the users of the target and all their attributes, such as privileges, access control properties or group memberships;
- the organizational units of the target used to organize the resources;
- the computers that are part of the target and their attributes, such as operating system, role of the machine, exposed network services.

As shown in Figure 7.2 the data acquisition phase can be performed in several ways (or modes), thus making the framework very flexible. These modes refer to:

- manual mode: data describing the network configuration is provided manually to the system through the use of configuration files;
- automatic mode: data describing the network is provided by means of automated tools using passive and active scanners;
- random mode: data describing the network is generated randomly. This mode can be used for testing or evaluation purposes.



(A) Automatic Mode

(B) Manual Mode

(C) Random Mode

FIGURE 7.2: Data Acquisition Modes

For example, "automatic mode" could consist of running a scanner to obtain the target data. On the contrary, the "manual mode" requires to insert data within a configuration file. Another interesting option is to use the data obtained from a randomly generated ("random mode") artificial network environment.

Let us remark that it is particularly challenging to obtain real data about enterprise environments since companies rarely release details about their internal organization. Hence, the generation of realistic artificial environments can be very beneficial for testing complex environments.

7.3 Network Knowledge Base

The data collected in the acquisition phase is organized in a structured knowledge base which keeps track of all the network data and configurations. This knowledge base works as an interface for further steps. This allows third-party developers to implement custom tools to collect data that can be used within the framework.

The collected data is organized according to the following entries:

- Domains: containing data about the domains analyzed and the trust relationships they may have with other domains;
- Users: containing the users and the information collected in the data acquisition phase;

- Computers: containing the computers within the target network and their attributes;
- Groups: containing information about the groups into which users are organized;
- Organizational Units, containing information about organizational hierarchies into which other elements of the network are organized;
- Access Control Lists, containing information about access control relationships between users, computers, domains, groups and organizational units.

The data stored within the knowledge base captures the hierarchical structure of the network organization as shown in Figure 7.3.

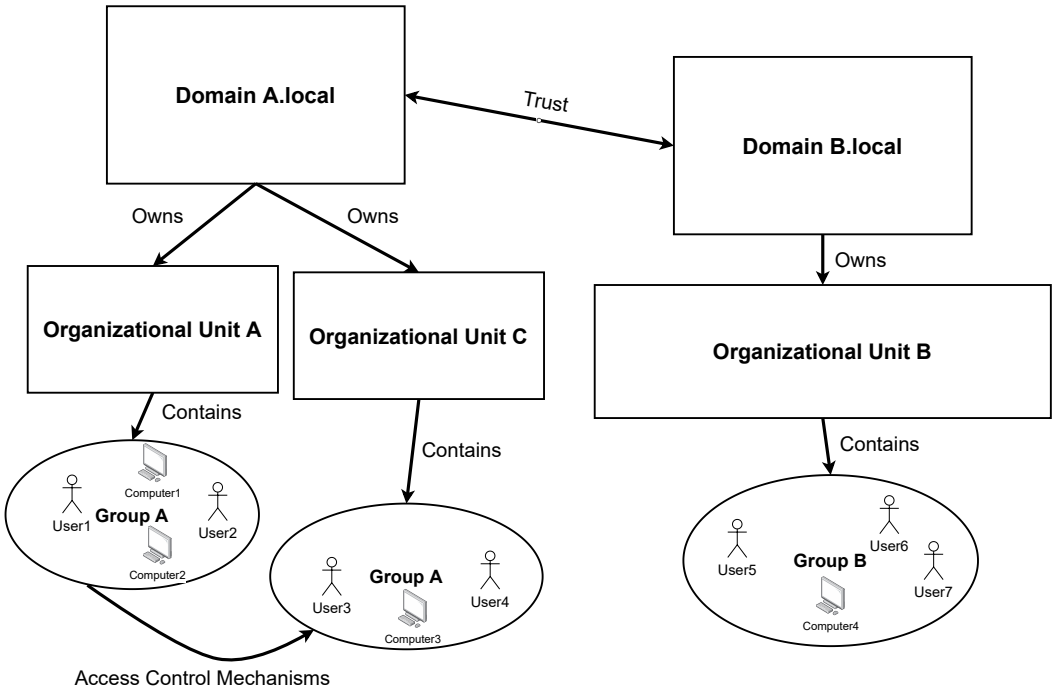


FIGURE 7.3: High-Level hierarchy of collected data

The data at this stage is organized in way that can be easily inserted into a graph database.

7.4 Network Graph Database

The data organized in a well structured knowledge base is stored within a non-relational graph database [155], [156]. Graph databases represent the state of the art of database technology since graph processing is behind numerous problems in computing. Nowadays these databases are employed in areas where information about data interconnectivity or topology is as important as the data itself, or even more important. In these applications, data and relations among the data, have the same priority.

In fact, graphs have the advantage of being able to store all the information about an entity in a single node and highlight information related to that node by edges that are connected to it.

This feature allows users to find hidden relationships which would be very difficult to highlight by using a different kind of database. In addition, graphs can nicely represent relationships in a computer network. As can be seen in the example graph depicted in Figure 7.4, users are often in some kind of relationship with other entities on a network and graphs are particularly good at capturing these relationships. The graph shown in Figure 7.4 shows user "John1" (in light green) belonging to two different groups (in yellow). One of these groups, namely "Remote Admins", has access to "Computer1" where the user "Joe22" has stored his credentials. The user "Joe22" belongs to the group of administrators. Basically the graph shows a path from the user "John1" to the group of administrators.

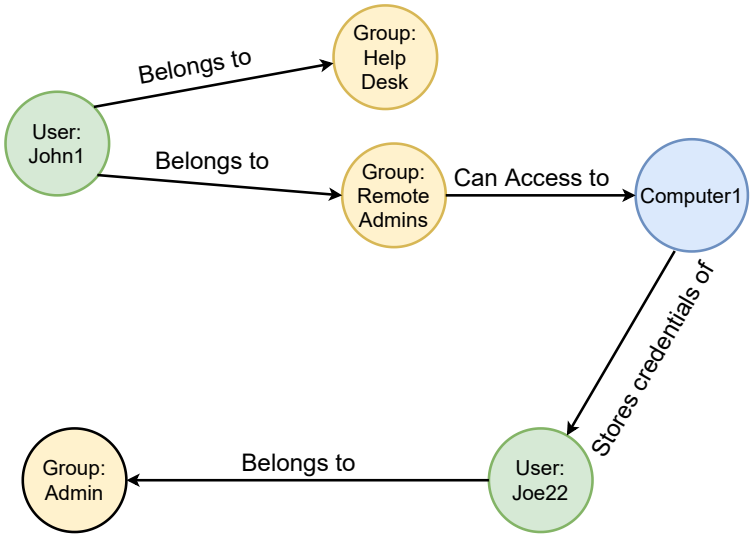


FIGURE 7.4: Example of a graph capturing network properties and relationships

In fact, graphs allow the construction of paths from users to domain administrators, that are the most privileged users within a domain. These paths are built from nodes that may represent entities on a network, such as users, computers, groups or organizational units and relationships that link the entities such as, "HasAccess","CanRDP","Contains","Owns".

The presence of nodes and relationships inside a graph database allows an easy identification of paths that could lead an attacker to the compromise of the entire network.

Note that, during this phase, a "compromised" user is specified. This models a scenario where an attacker has compromised that specific user. Shortest paths in this context highlight the capabilities that the attacker has on the network.

7.5 Information Extractor

Information extraction and feature engineering are of fundamental importance in AI-powered systems. In the context of this framework there is a need of finding useful security insights about the graph to be used by the classification systems. For this purpose, in this step, we take advantage of the domain knowledge about computer systems security, attacks and vulnerabilities to identify paths, attributes, and characteristics of a graph useful to determine whether an enterprise environment is vulnerable or safe.

The information extraction involves the computation of overall scores associated to each node in the graph. These scores assess the degree of vulnerability associated with the nodes and their value depends on the interest that an attacker may have for a specific node.

Computing overall scores allow the identification of weighted shortest paths, that is, paths that may be more "interesting" for an attacker and that do not directly depend merely on the number of hops between an attacker and its target.

Once the overall scores have been assigned to each node in the graph, the information extraction phase involves the extraction of two sub-graphs. These sub-graphs correspond to all the shortest paths and a weighted shortest path from an attacker node and a target node.

Note that the non-weighted shortest paths prioritize the number of hops between two end nodes, while the weighted shortest path takes into account the overall scores, hence, prioritizes the network characteristics that may lead an attacker to compromise the network. The identification of the weighted path relies on the Dijkstra's algorithm.

Taking into account these two types of shortest paths, a general set of attributes have been identified. These attributes are general enough to be applied to very diverse networked environments.

The set of attributes contains insights about the structure and the properties of both the non-weighted and the weighted shortest paths, namely:

- number of "Access Control relationships", that is, the rights and permissions a node has on other nodes;
- number of "Remote Access relationships", that is, the remote control capabilities users have on specific machines, such as SSH, RDP and others;
- number of "Hierarchical relationships", that is, the groups and hierarchical organizational units within the paths;
- number of "Critical relationships", that is, features that are critical from the perspective of security, such as number of user sessions on machines or administrative relationships between nodes;
- features related to the structure of the shortest paths, such as number of nodes, number of relationships, number of shortest paths, sum of the overall scores related to the nodes, number of old operating systems within the path.

These attributes appropriately capture the status of the security of an enterprise environment from the perspective of a compromised user.

7.6 Machine Learning Classifier

Machine learning is particularly useful in applications where heuristics-based systems would be overly complicated. Within this framework the classifier is based on a standard supervised machine learning pipeline as shown in Figure 7.5.

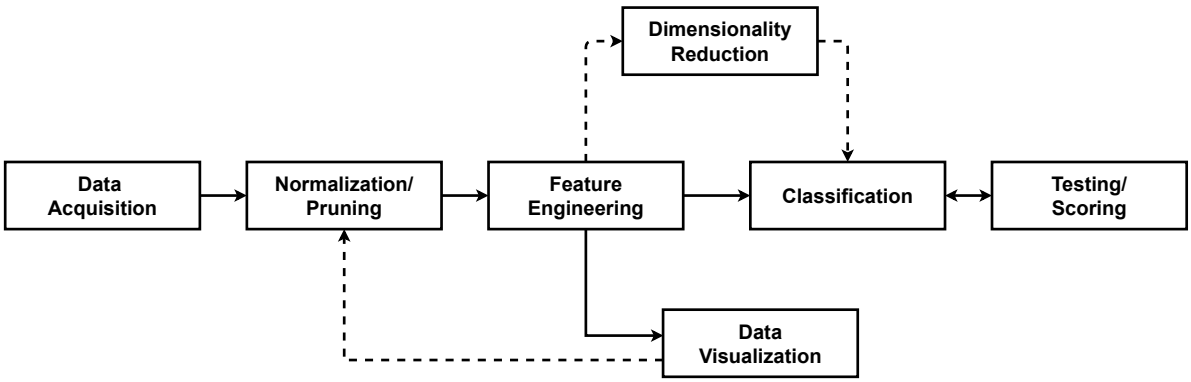


FIGURE 7.5: Supervised machine learning pipeline used within the framework

Data preprocessing is a fundamental step in a machine learning system. Data gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations or missing values. Analyzing data which has not been carefully checked and corrected or discarded can lead to completely misleading conclusions.

If information in a dataset is redundant, ambiguous, unreliable, or noisy, then finding patterns during the training step is significantly more difficult.

In the context of this thesis work, the data preprocessing step was very limited, since most of the data was artificially generated, hence, the only actions to perform were related to the removal of duplicates.

The output of a data preprocessing system is the final training set.

One of the most critical steps of the workflow is represented by feature engineering, that is, the process of using domain specific knowledge to select appropriate features for machine learning algorithms to work. A feature is a characteristic of the phenomenon under investigation. Feature engineering is both difficult and expensive. Let us remark that an incorrect selection of features can lead to poor classification performance.

Data visualization is another important step of the proposed workflow. This step is applied to better understand data and detect anomalies (e.g., outliers) which have to be pruned.

A supervised learning algorithm is applied to train a model for classification or regression. Finally it is necessary to evaluate the system, that is, testing and scoring phase. The evaluation relies on different metrics, e.g., accuracy for balanced datasets, precision, recall and F1-score for unbalanced

datasets. In case of unsatisfactory results, the process is iterated with the objective of improving the system performance.

Note that evaluation of the results is a critical step in machine learning pipelines. When the amount of data is significant a classical three-fold approach is sufficient. This approach involves splitting the entire dataset into three partitions: the training set (used for training the model), the cross-validation set (used for tuning the model parameters) and the testing set for evaluating the model. This approach does not represent the best choice when data availability is a problem. In fact, the k-fold approach for validation is a better choice when dealing with small datasets [157].

In this work, a k-fold approach for validation is suggested. The k-fold approach in these contexts has many benefits, since enterprise network data is difficult to retrieve, and its labeling other than being time-consuming requires domain expertise. The final result of the classifier in our context determines whether the provided input network is vulnerable or safe.

Chapter 8

Implementation of the Framework

This chapter presents the solutions proposed in this thesis work for the implementation of the framework for automatic AI-assisted security assessment of enterprise networks. More precisely, we describe the design of the tools developed for automating the enumeration process. Since the target enterprise networks are based on Active Directory, we focus on the enumeration tools both from a domain-joined Microsoft Windows machine and from a non-domain joined machine. In addition, we present the graph-based approach and the details of the nodes and relationships used for automatic threat modeling. Finally, we describe the design of a generator of artificial Active Directory environments and the approach for classifying the graph models.

8.1 Active Directory Enumeration

In penetration testing activities, "enumeration" is the process consisting of the collection of information about the target technological infrastructure for discovering potential vulnerabilities or misconfigurations. It is possible to enumerate a system using sniffers and scanners or other tools that automate this process. During this thesis work two tools used for enumeration of enterprise Active Directory environments have been designed and developed. These tools can perform the enumeration from both domain-joined Microsoft Windows computers and non-domain joined computers. In fact, since Active Directory is a technology which naturally fits Microsoft systems, some features are only available on machines running Microsoft Windows operating systems.

Both tools provide a report in a JSON format containing detailed information about the target domains such as security principals, group memberships, trust relationships, organizational units, and group policies.

In detail, the tool dedicated to the enumeration from machines that are joined to a Microsoft domain (called WinForestMap) is developed using PowerShell to take advantage of the features offered by Microsoft Windows API, COM, WMI, and the .NET library. This tool relies on PowerView and the native Active Directory Module.

As shown in Figure 8.1, the architecture of WinForestMap consists of four main components including a general "main" controller whose goal is to

drive other operations, a PowerView based module and an AD (Official Microsoft module available for Active Directory administration) based module that collect information about the target infrastructure through DNS and LDAP queries and ADReport module that converts all the gathered information into a structured JSON component.

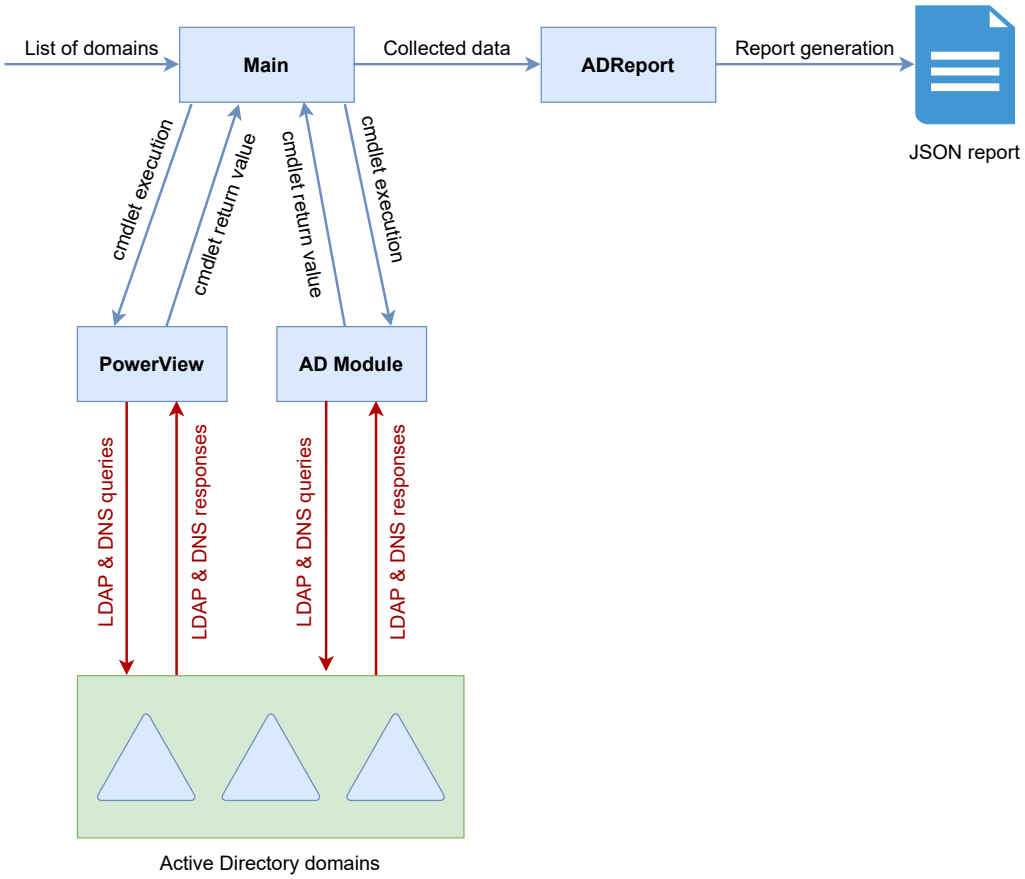


FIGURE 8.1: WinForestMap architecture

The `Main` component receives as input the list of domains to be enumerated and executes the cmdlets (PowerShell user defined functions) contained in PowerView and in the Microsoft Active Directory Module. These cmdlets interact with the forest's Domain Controllers using the LDAP and the DNS protocols. These protocols make it possible to collect a significant amount of information about the directory objects. The `PowerView` module provides the most used cmdlets for the enumeration process. For example, `Get-NetUser`, `Get-NetGroup`, and `Get-NetComputer` are used for retrieving the lists of users, groups, and computers, respectively. In addition, this module provides the cmdlets to obtain more specific information, such as the IP addresses of the domain computers (`Get-IPAddress`) or the list of computers with unconstrained delegation enabled (`Get-NetComputer -Unconstrained`).
The `ADModule` provides additional enumeration capabilities for collecting domain's properties. At the end of the enumeration process, the `Main` component sends the collected data to the `ADReport` module responsible for generating a report in a JSON format.

By running WinForestMap as a non-privileged user, it is possible to retrieve information about a forest, such as forest domains, domain policies, Kerberos policies, security principals, computers, groups, OUs, GPOs, ACLs, SMB shares, domain trusts, and active user sessions. Additional information about SMB shares can be obtained by running the tool from an account with elevated privileges.

The second tool designed and developed, namely ForestMap, is dedicated to the enumeration process from machines running Linux or MacOS operating systems and not belonging to the target Active Directory environment. As shown in Figure 8.2, the architecture of ForestMap consists of six components including a general "main" controller whose goal is to drive other operations and generate a final structured JSON report, and multiple secondary tools that are able to collect information using different protocols such as DNS, LDAP or ICMP.

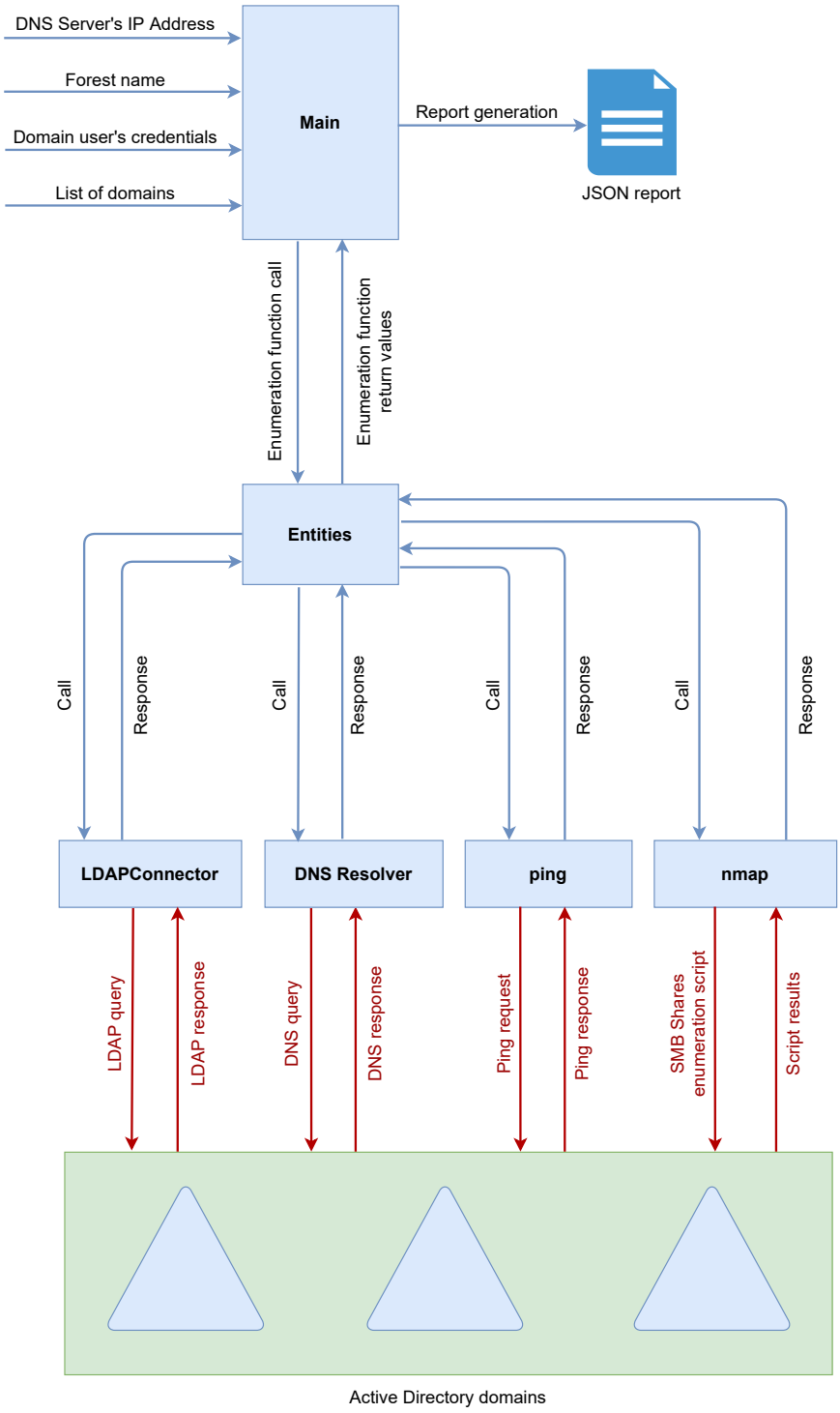


FIGURE 8.2: ForestMap architecture

Starting from features, such as the IP address of the Domain Controller acting as DNS server, the name of the target forest, the credentials of a compromised account, and the target domains, the `Main` component is responsible for calling the enumeration functions provided by the `Entities` module. This module in turn interacts with the `LDAPConnector`, the `DNS Resolver`, `ping`, and `nmap` for collecting information about the Active Directory environment. In detail, the `LDAPConnector` component relies on OpenLDAP to

execute LDAP queries using the credentials of a compromised user and retrieve information about the directory objects. The `DNS Resolver` component performs DNS queries for translating the hostnames of computers and Domain Controllers into their IP addresses. The `ping` command is used for checking the availability of domain computers, while `nmap` is exploited for collecting information about the SMB shares of the current domain. At the end of the enumeration process, the `Main` component generates a JSON report to be analyzed by threat modeling tools.

Table 8.1 details the output information contained in the structured JSON modules extracted from LDAP queries. More precisely, the table summarizes the information collected on a test target domain called `CARONTE.LOCAL`, through LDAP queries. The table also shows the search base, that is, the LDAP path used to start the search, and the filters applied.

Information	**Search base**	**Search filter**
List of computers	`DC=CARONTE,DC=LOCAL`	`objectClass=Computer`
List of computers with unconstrained delegation	`DC=CARONTE,DC=LOCAL`	`(&(objectCategory=computer)(objectClass=computer)(userAccountControl:1.2.840.113556.1.4.803:=524288))`
List of domains	`CN=PARTITIONS, CN=CONFIGURATION, DC=CARONTE,DC=LOCAL`	`NETBIOSName=*`
List of Domain Controllers	`OU=DOMAIN CONTROLLERS, DC=CARONTE,DC=LOCAL`	`objectClass=Computer`
List of GPOs	`DC=CARONTE,DC=LOCAL`	`objectClass=groupPolicyContainer`
List of groups	`DC=CARONTE,DC=LOCAL`	`objectClass=Group`
List of OUs	`DC=CARONTE,DC=LOCAL`	`objectClass=OrganizationalUnit`
List of users	`DC=CARONTE,DC=LOCAL`	`(&(objectCategory=person)(objectClass=user))`
List of users with unconstrained delegation	`DC=CARONTE,DC=LOCAL`	`(&(&(objectCategory=person)(objectClass=user))(userAccountControl:1.2.840.113556.1.4.803:=524288))`
List of kerberoastable users	`DC=CARONTE,DC=LOCAL`	`(&(&(servicePrincipalName=*)(UserAccountControl:1.2.840.113556.1.4.803:=512))(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))`

TABLE 8.1: Information about the domain objects retrieved using LDAP queries.

In summary, WinForestMap and ForestMap represent good options for the automation of a detailed enumeration phase aimed at collecting information about a target and storing this information into structured JSON reports. These JSON reports are designed in order to be easily parsable, converted to a graph and stored in appropriate databases as described in what follows.

8.2 Active Directory Threat Modeling

As already discussed, threat modeling is the process following the enumeration phase applied with the aim of identifying, classifying, and prioritizing the security risks that affect a system. This step of penetration testing is crucial for identifying the "best" paths to be followed to compromise a target Active Directory environment.

In this context, a graph-based tool, namely ADGraphGenerator, for the generation of an Active Directory graph model starting from the report produced enumeration tools has been designed and developed. The generated

graph model contains nodes and edges. Nodes represent domains, users, computers, groups, Organizational Units, and GPOs, while edges represent relationships between nodes (e.g., `MemberOf`, `Contains`, `GenericAll`, `TrustedBy`). After the generation of the graph, the tool assigns to each node a `criticalasset` boolean property, to specify that the asset is considered critical within the infrastructure. In addition, each node is assigned a score that depends on its properties and each edge a cost.

Figure 8.3 shows the architecture of `ADGraphGenerator` where we can identify three core components, responsible for reading the normalized input data, computing the scores and costs associated with nodes and edges and interacting with the Neo4J graph database.

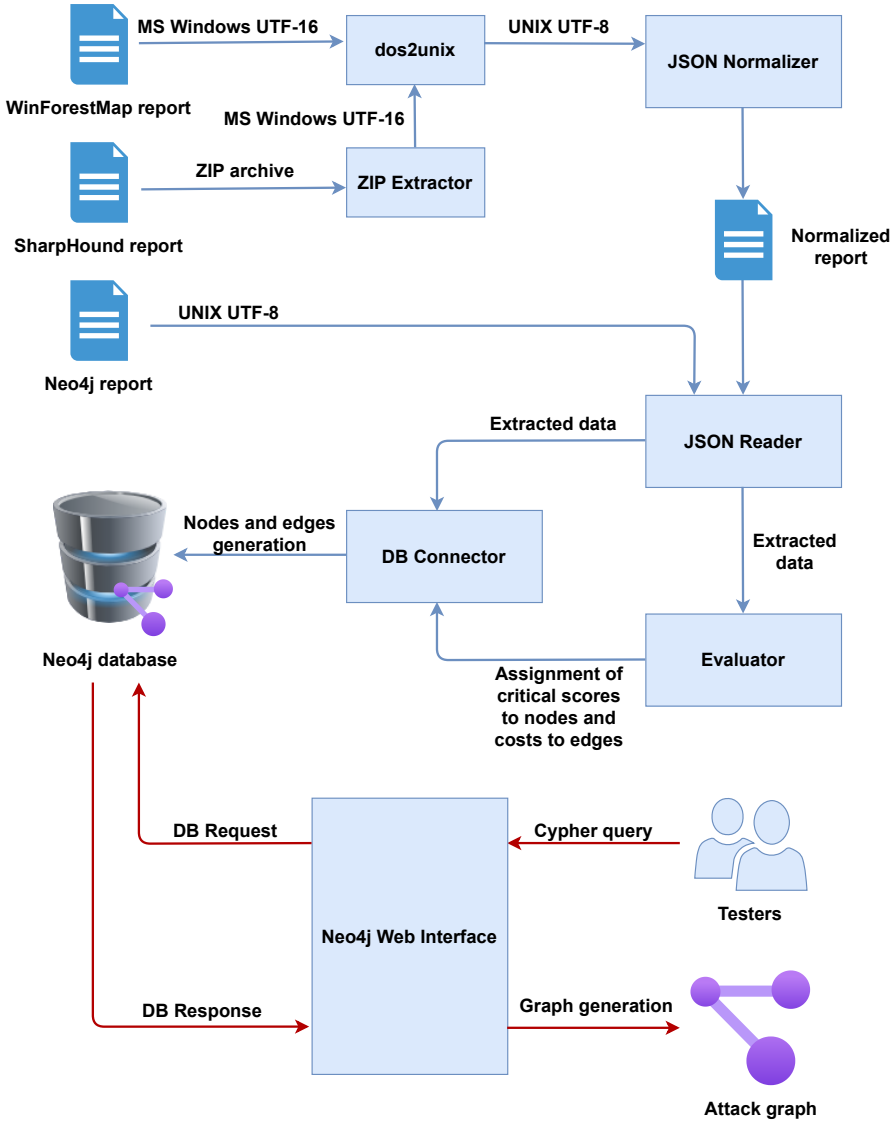


FIGURE 8.3: ADGraphGenerator architecture

Note that, in order for the tool to be flexible enough, different types of reports are supported. In fact, the `JSON Normalizer` component is dedicated to this purpose. More precisely, the `ZIP Extractor` component extracts the

JSON files from the ZIP archive generated by SharpHound, that is, a well-known tool used in Active Directory enumeration, this has been done for compatibility reasons with common third-party tools. The `JSON Normalizer` component normalizes the reports by adding missing properties, converting `null` values to strings, rewriting property values in the appropriate formats, and removing information not being used.

The `JSON Reader` component takes as input the normalized JSON report and calls the methods of the `DB Connector` component for storing nodes and edges with their properties in a Neo4j database instance. The `Evaluator` component processes the data and identifies critical assets (i.e., domain nodes, privileged users and groups, Domain Controllers), and computes overall scores for the nodes and costs for the edges. Finally, the `Neo4j Web Interface` processes the Cypher queries and generates the requested attack graphs.

Note that, the choice of a graph database, that represents the state of the art of non-relational databases, was driven by their performance and suitability to store data that can be well described as a graph and their ability at identifying hidden relationships among data.

8.2.1 Overall Scores

As previously mentioned, the most important feature provided by ADGraph-Generator is the identification of the most vulnerable nodes within an Active Directory environment by means of a score assigned to each node according to its properties and relationships. Various properties and relationships of the nodes are considered for assigning these scores, such as version of operating system, password policies, access control lists. The overall score of a node N is calculated by assigning a score to each property p_i associated with that node and to each outgoing relationship r_j belonging to the same node according to their exploitability in a potential attack. Thus:

$$overallscore(N) = \sum_i score_{p_i} + \sum_j score_{r_j}$$

Table 8.2 summarizes the scores assigned to the properties and to the relationships considered to compute the overall score of each type of node.

Object Type	Property		Relationship	
	Name	Value	Name	Value
Computer	is a DC	10	MemberOf	5-30
	operatingsystem	0-30	AdminTo	2
	lastlogon	1	HasSession	2
	unconstraineddelegation	5	CanRDP CanPSRemote ExecuteDCOM	2
			AllowedToAct AllowedToDelegate	5
			ACLs	5-10
Domain			TrustedBy	0-5
			ACLs	10-30
GPO			ACLs	5-10
Group	is a privileged group	20-50	MemberOf	20-50
			AdminTo	10
			CanRDP CanPSRemote ExecuteDCOM	2
			ACLs	5-30
OU			ACLs	5-10
User	admincount	5	MemberOf	20-50
	description	1-5	AdminTo	10
	dontreqpreauth	5	HasSession	3
	pwdneverexpires	3	CanRDP CanPSRemote ExecuteDCOM	5
	hasspn	5	AllowedToAct AllowedToDelegate	5
	unconstraineddelegation	5	ACLs	5-30
	passwordnotreqd	10		
	pwdlastset	3		
	lastlogon	1		

TABLE 8.2: Default scores of properties and relationships for the computation of the overall scores

It is worth mentioning that the score assignment requires solid domain knowledge about the kinds of attacks affecting enterprise infrastructures and their severity. As a simple example, scores of computers running with an operating system no longer supported by Microsoft are higher with respect to scores of computers with a modern patched operating system. Similarly, the score of a computer running the Microsoft Windows XP operating system and with two active user sessions can be very high, namely 34 in our example.

Once the overall scores have been computed for all the nodes of the graph, the tool computes the cost associated with each edge, that is:

$$cost(r_{NM}) = \log\left(1 + \frac{1}{overallscore(M) + 1}\right)$$

where r_{NM} denotes the relationship associated with the edge connecting the node N to the node M.

The logarithm is used to remove the skewness of the data and one is added to the argument of the logarithm to obtain positive costs. Finally, one is added to the overall score of the node M to avoid a zero division error.

The cost was chosen inversely proportional to the overall score of the node M in order to favor the relationships that lead to the most critical nodes when the shortest path is computed with Dijkstra's algorithm [118]. We recall that this algorithm is used to find the shortest path from an initial node to the destination node in a weighted directed graph.

In our case, the Dijkstra's algorithm is applied to compute the total cost of each path, that is, the sum of the costs of the edges traversed to reach the target node starting from an initial node and identify the path with the lowest total cost.

8.3 Simulation of Active Directory Domains

Simulated environments are very useful to carry out experiments. In fact, companies and organizations rarely agree on sharing information about their technological infrastructures and in particular of Active Directory environments. To ease the experimental activities, during this thesis a tool for the simulation of Active Directory domains has been designed and developed. The tool, named `ADRandom`, allows the random generation of realistic graph models representing the relationships between the entities of a domain.

The graph models are generated according to a list of specifications, such as number of nodes and number of trusts. Figure 8.4 shows the architecture of this tool.

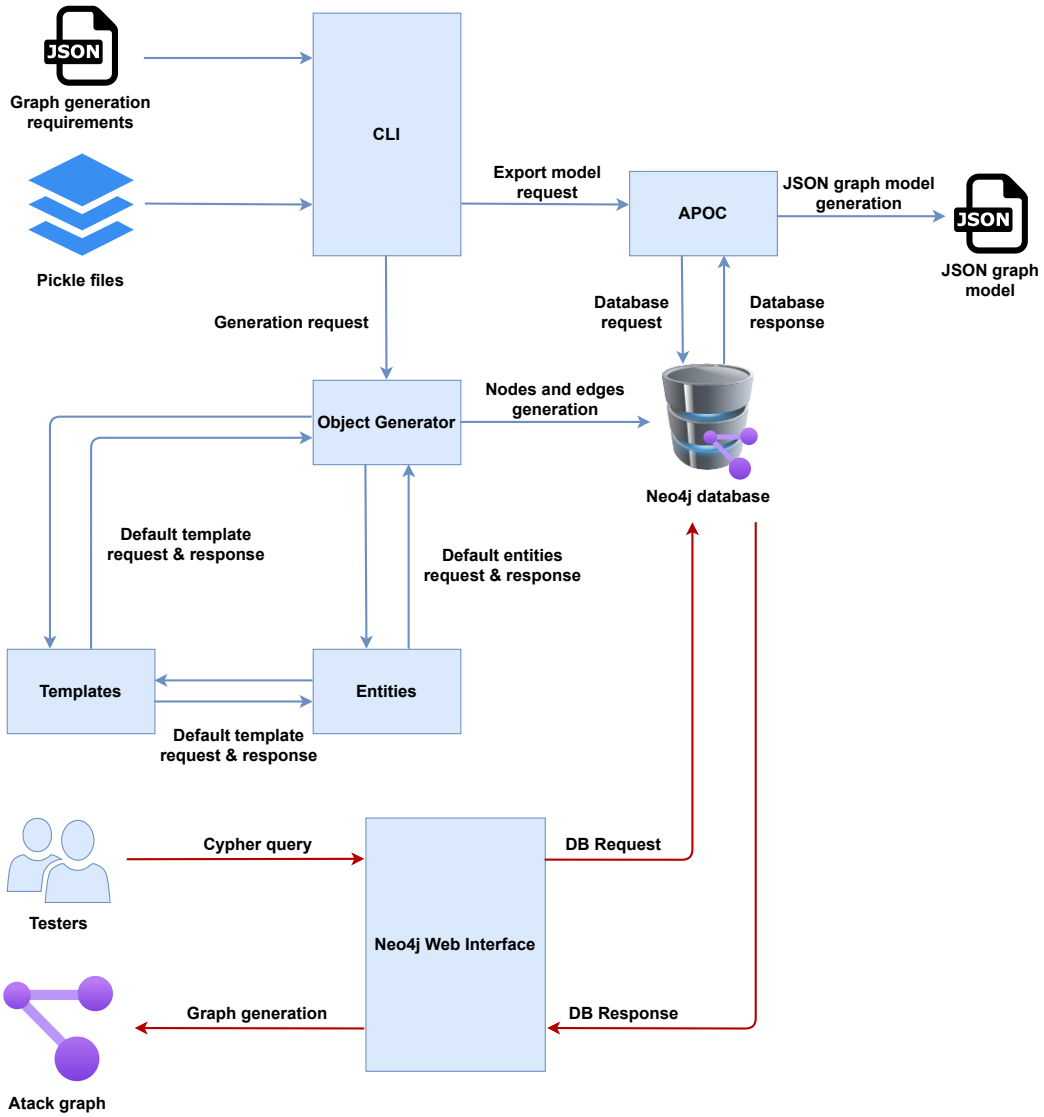


FIGURE 8.4: ADRandom architecture

Starting from the specifications of the graph model, the `CLI` component calls the methods provided by the `Object Generator` component for creating Active Directory objects and their relationships. In addition, the `CLI` component loads the pickle files that contain the first names, the last names, and the domain names to be used for the generation of the users and of the domain trusts. The `Entities` and the `Templates` components. In details, the `Entities` component provides the methods for the generation of the default groups (e.g., `Administrators`, `Domain Users`), users (e.g., `Administrator`, `Guest`) and Access Control Lists, while the `Templates` component provides basic templates used to generate objects and relationships that allows the generation of realistic Active Directory environments. The generated objects are then stored in a Neo4J database instance. Furthermore, an add-on of the `CLI` component is the `APOC` plugin for exporting the generated graph model into a JSON file. Finally, the `Neo4j Web Interface` processes the Cypher queries for analyzing the graph model.

The ability to generate graphs according to the specifications is the most relevant functionality offered by `ADRandom` to obtain models representative of real Active Directory environments. For example, it is possible to choose the size of the Active Directory environments by setting the number of domain trusts, Security Principals, computers, OUs, groups, and GPOs. Moreover, it is possible to choose the probabilities associated with object properties.

It is also possible to generate environments affected by specific vulnerabilities (e.g., various types of ACL misconfigurations, inadequate security policies) that make them possible targets of cyber-attacks, such as AS-REP Roasting and DCSync. Table 8.3 summarizes the most relevant parameters used in the generation of an Active Directory domain.

Object	Property	Description	Value
ACL	`ACLsProbability`	Probability of each access control right (e.g., `GenericAll`, `AddMember`, `WriteDacl`).	Integer for each right (0-100). The sum of the probabilities must be equal to 100.
Computer	`nComputers`	Number of computers.	Integer >0
Computer	`CanRDPFromUserPercentage` `CanRDPFromGroupPercentage`	Maximum percentage of computers with `CanRDP` edges from users/ groups.	Integer (0-100)
Computer	`CanPSRemoteFromUserPercentage` `CanPSRemoteFromGroupPercentage`	Maximum percentage of computers with `CanPSRemote` edges from users/ groups.	Integer (0-100)
Computer	`ExecuteDCOMFromUserPercentage` `ExecuteDCOMFromGroupPercentage`	Maximum Percentage of computers with `ExecuteDCOM` edges from users/ groups.	Integer (0-100)
Computer	`AllowedToDelegateFromUserPercentage` `AllowedToDelegateFromComputerPercentage`	Maximum percentage of users/ computers with `AllowedToDelegate` edges.	Integer (0-100)
Computer DC User	`enabled`	Probability that an object is enabled.	Integer (0-100)
Computer User	`unconstraineddelegation`	Probability that an object has unconstrained delegation enabled.	Integer (0-100)
Computer DC	`osProbability`	Probability of each OS version.	Integer for each OS version (0-100). The sum of the probabilities must be equal to 100.
Domain	`functionalLevelProbability`	Probability of each functional level value.	Integer for each functional level (0-100). The sum of the probabilities must be equal to 100.
Domain	`Trusts`	Number of Inbound, Outbound and Bidirectional trusts.	Integer >= 0
GPO	`nGPOs`	Number of GPOs.	Integer >0
Group	`nGroups`	The number of groups.	Integer >0
OU	`nOUs`	Number of OUs.	Even integer >0
User	`nUsers`	Number of users.	Integer >0
User	`dontreqpreauth`	Probability that a user has Kerberos pre-authentication disabled.	Integer (0-100)
User	`hassp`	Probability that a user has a SPN.	Integer (0-100)
User	`passwordnotreqd`	Probability that a user does not have a login password.	Integer (0-100)
User	`pwdneverexpires`	Probability that a user's password never expires.	Integer (0-100)
User	`sidhistory`	Probability that a user previously belonged to another domain.	Integer (0-100)

TABLE 8.3: Most relevant parameters used for the generation of graph models.

8.3.1 Active Directory Domain Generation

In what follows, we describe the generation of an Active Directory domain and we discuss the most relevant relationships inside the domain. The example refers to a domain consisting of 30 users and 20 computers that are contained (`Contains` relationship) in 10 Organizational Units. In addition, the domain has to include an Organizational Unit containing six Domain Controllers and a parent-child relationship with another domain has to be considered. Table 8.4 summarizes some specifications used to generate this model.

Object	**Property**	**Value**
Computer	`nComputers`	20
Computer	`CanRDPFromUserPercentage`	10
Computer	`CanRDPFromGroupPercentage`	10
Domain	`Trusts`	`Inbound`: 0, `Outbound`: 0, `Bidirectional`: 1
Group	`nGroups`	10
OU	`nOUs`	10
User	`nUsers`	30

TABLE 8.4: Specifications used for the generation of the graph model

Basically, `ADRandom` takes as input data like the one shown in Table 8.4 and by interacting with the Neo4J instance fills the database with the generated Active Directory environment.

Figure 8.5 shows the parent-child trust relationship between the `TESTLAB.LOCAL` and the `BACKET.TESTLAB.LOCAL` domains.

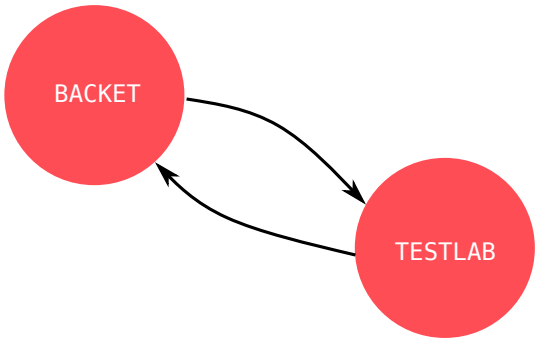


FIGURE 8.5: Graph showing `TrustedBy` relationships between two domains

The graph of Figure 8.6 shows the generation of users and computers by and how these are hierarchically organized within Organizational Units that belong to the domain. In particular, yellow nodes in this graph represent users, while purple nodes represent computers, and blue nodes are Organizational Units. The green node denotes the compromised user who was

exploited for enumerating the Active Directory domain represented by the graph model. Users and computers are contained in Organizational Units and Organizational Units are contained in the domain (the central red node).

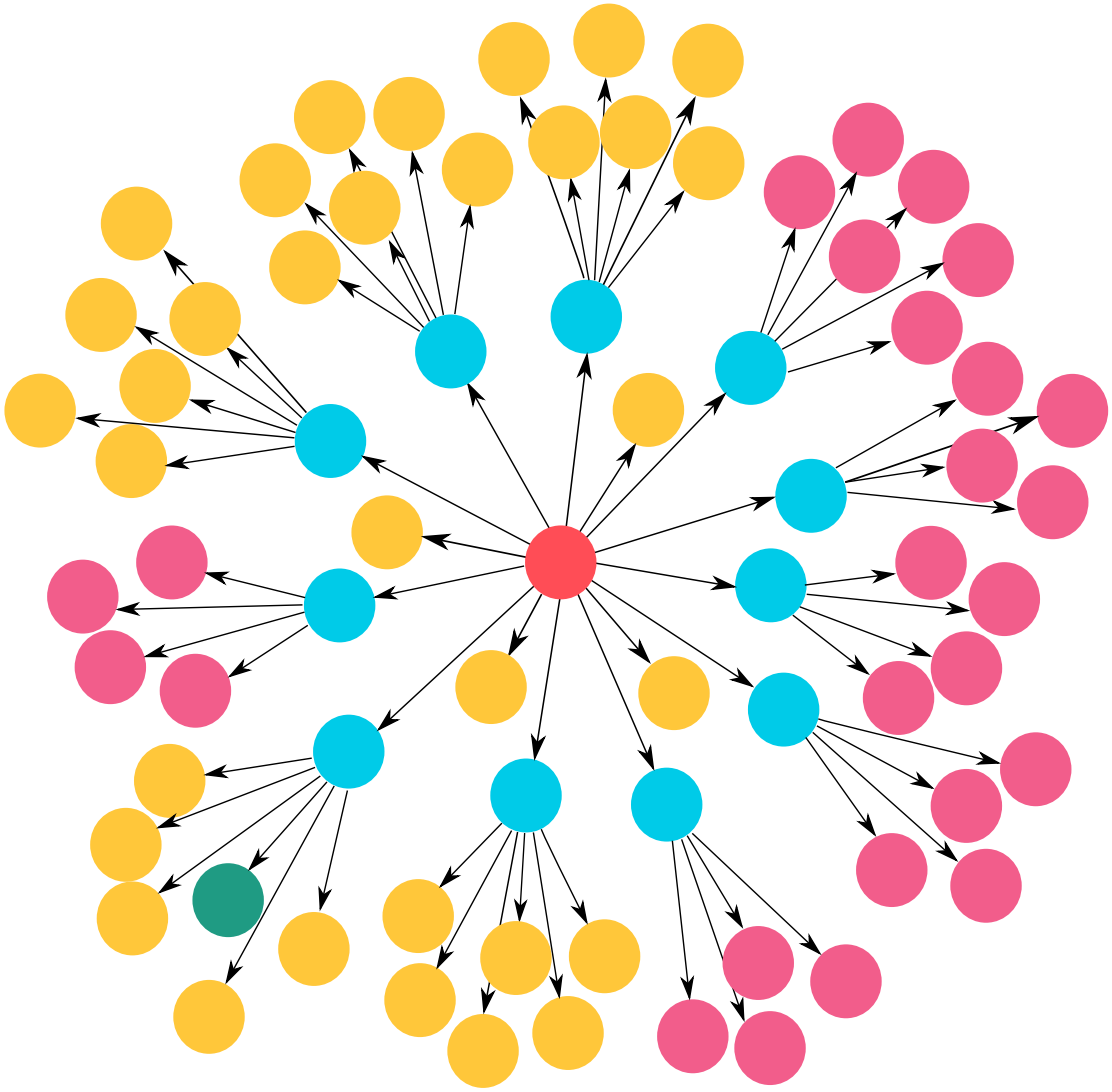


FIGURE 8.6: Example of a graph showing `Contains` relationships between objects. Yellow nodes correspond to users, red to computers, cyan to OUs, green to the compromised user and red to the domain

The graph in Figure 8.7 shows the users and the computers that are members (`MemberOf` relationship) of the domain groups (depicted in orange). Purple nodes on the top left are the computers that belong to the `DOMAIN COMPUTERS` group, while yellow nodes and the green one are the users that belong to the `DOMAIN USERS` group. Purple nodes on the right are the Domain Controllers that belong to the `DOMAIN CONTROLLERS` and to the `ENTERPRISE DOMAIN CONTROLLERS` groups.

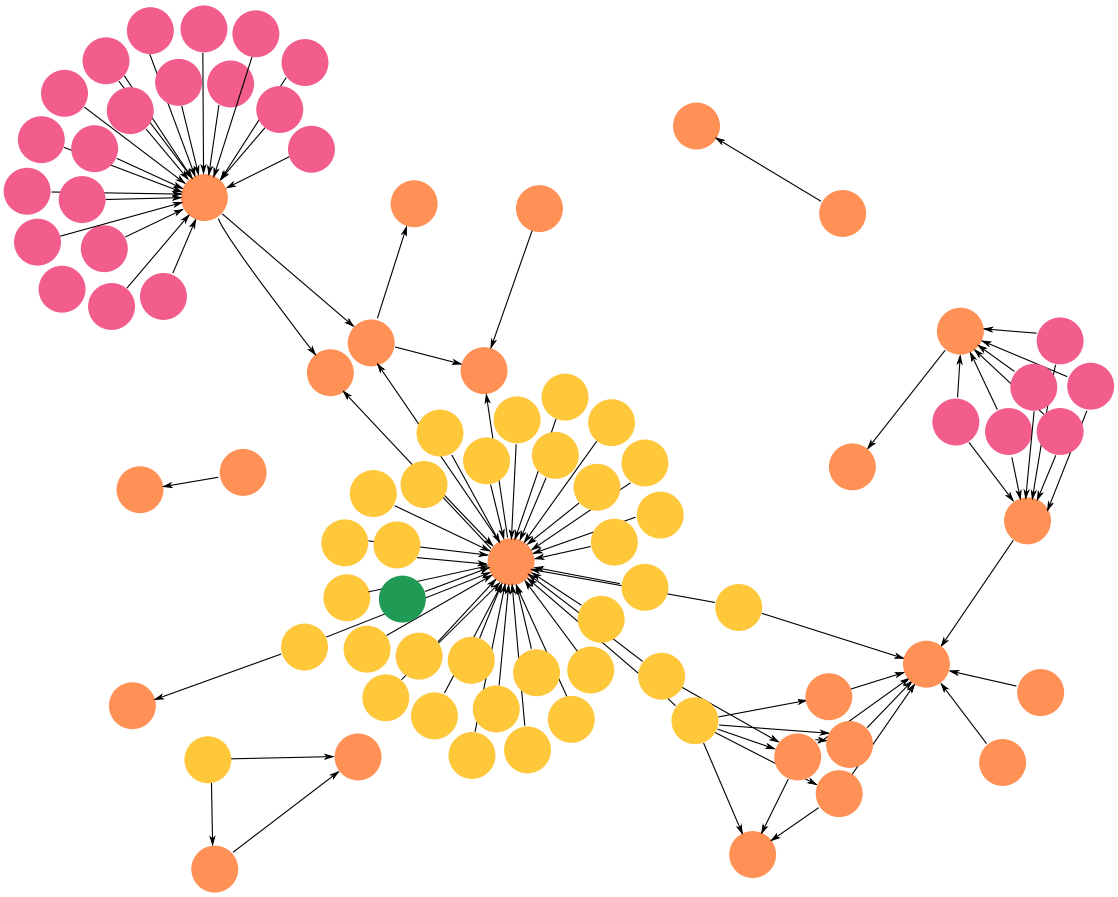


FIGURE 8.7: Graph showing `MemberOf` relationships between objects. Yellow nodes correspond to users, red to computers, green to the compromised user and orange to groups

The graph in Figure 8.8 shows the Security Principals that can open a session on a remote computer using the Remote Desktop Protocol. In this case, two computers can be remotely controlled by a user and two computers can be controlled by the members of a group. In fact, the `CanRDPFromUserPercentage` and the `CanRDPFromGroupPercentage` parameters were set equal to 10% (see Table) 8.4.

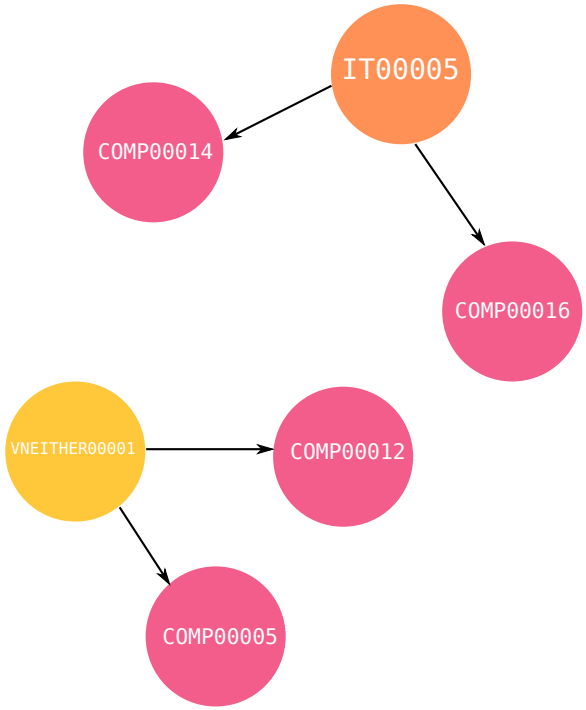


FIGURE 8.8: graph showing CanRDP relationships between security principals. Yellow nodes correspond to users, red to computers and orange to groups

8.4 Machine Learning Classifier

To automatically classify enterprise networks a supervised learning approach has been used. For this purpose, different classifiers have been tested for the proposed methodological framework. All of these classifiers have been validated through the use a k-fold with $k = 10$ validation approach. This has been done in order to tackle the relatively small amount of available data. In fact, organizations and companies rarely share information about their technological infrastructure.

In general, classifiers require an accurate feature engineering phase, since the performance of these systems mainly depend on the choice of the features. In addition, due to issues related to the curse of dimensionality, on small datasets it is not beneficial to use a large number of features. For these reasons, features have to accurately be identified. The proposed classification system takes advantage of the properties related to the shortest paths between the compromised user and the target administrators group. Starting from these shortest paths, features are extracted. These features are statistically analyzed with the purpose of removing the highly correlated ones, since these would deteriorate the performance of the overall system. Finally, different classification algorithms such as logistic regression, support-vector machines and random forests have been tested to perform the classification task.

Chapter 9

Experimental Results

This chapter presents an experimental application of the proposed framework to detect vulnerable enterprise networks. In particular, the chapter offers an overview of the dataset used to train, tune and validate the classifier. Moreover, classification results are discussed in detail.

9.1 Graph Dataset

The dataset used in these experiments consists of artificial graph models generated using the `ADRandom` tool. The characteristics of the models, such as the number of users, the number of computers or groups, are drawn from a uniform distribution specified through the settings provided to `ADRandom`.

Note that we are ultimately interested in identifying and analyzing shortest paths, but before being able to retrieve shortest paths we have to generate realistic graph models from which the shortest paths are extracted. For this purpose, we generated 220 graphs by using `ADRandom` tuned with the settings shown in Table 9.1.

Property	**Min**	**Max**
Users	50	150
Computers	50	150
Groups	20	100
OUs	39	41
GPOs	20	40
Domains	2	6

TABLE 9.1: Settings of the number of entities used for graph generation

Basic statistics related to characteristics of the generated graphs are presented in Table 9.2

Property	Mean	Std Dev
Users	103	28
Computers	120	29
Groups	113	24
OUs	41	1
GPOs	31	6
Domains	4	1
Number of Total Nodes	415	47
Total Relationships	4124	590

TABLE 9.2: Basic statistics of the graphs generated by ADRandom

All the provided statistics were rounded to the closest integer.

Let us remark that, although these settings are defined randomly through a uniform distribution, by following a set of heuristics `ADRandom` is able to determine whether the resulting graph represents a realistic environment. Moreover a set of builtin properties has to be checked.

For example, the generation of an Active Directory environment with 10 groups, will include additional 54 that have to be added to the specified ones, since every Active Directory environment has by default 54 builtin groups.

For each of the Active Directory graphs, we extract two sub-graphs, that is, one related to all the shortest paths and the other related to the single weighted shortest path identified by using the Dijkstra's algorithm. Let us remark that these paths are computed starting from a compromised user whose target is the `Domain Administrators` group.

The features extracted from these sub-graphs refer to:

- ACE relationships: total number of relationships related to access control mechanisms, such as `GenericWrite`, or `GenericAll`;
- Group nodes: total number of nodes representing hierarchical grouping, such as `nGroup`, or `nOUs`;
- Remote relationships: total number of relationships related to remote access capabilities, such as `CanRDP`, or `CanPSRemote`;
- Critical relationships: total number of relationships considered "Critical" from a security point of view, such as `AddMember`, `HasSession`, `AdminTo`;
- Old Machines: total number of old machines in the paths;
- Critical Assets: total number of critical assets in the paths;
- Nodes: total number of nodes in the paths;
- Relationships: total number of relationships in the paths;
- Critical Score: overall score corresponding to the shortest path;

- Path Cost: overall cost associated with the shortest path;
- Shortest Paths: number of shortest paths from the compromised user to the target;

Note that for each of these features (except for the number of Shortest Paths) we must consider a dual feature corresponding to the weighted shortest path.

The distributions of the features extracted from the shortest paths are summarized in Figures 9.1 and 9.2.

As can be seen from Figure 9.2 (K), the number of different shortest paths varies between 1 and 35 with an average of 5 shortest paths per graph. In Figure 9.1 (A) we can notice how the number of nodes on the weighted shortest path tend to be around 5 and 6 suggesting that this is a typically length for a shortest path within the considered graphs.

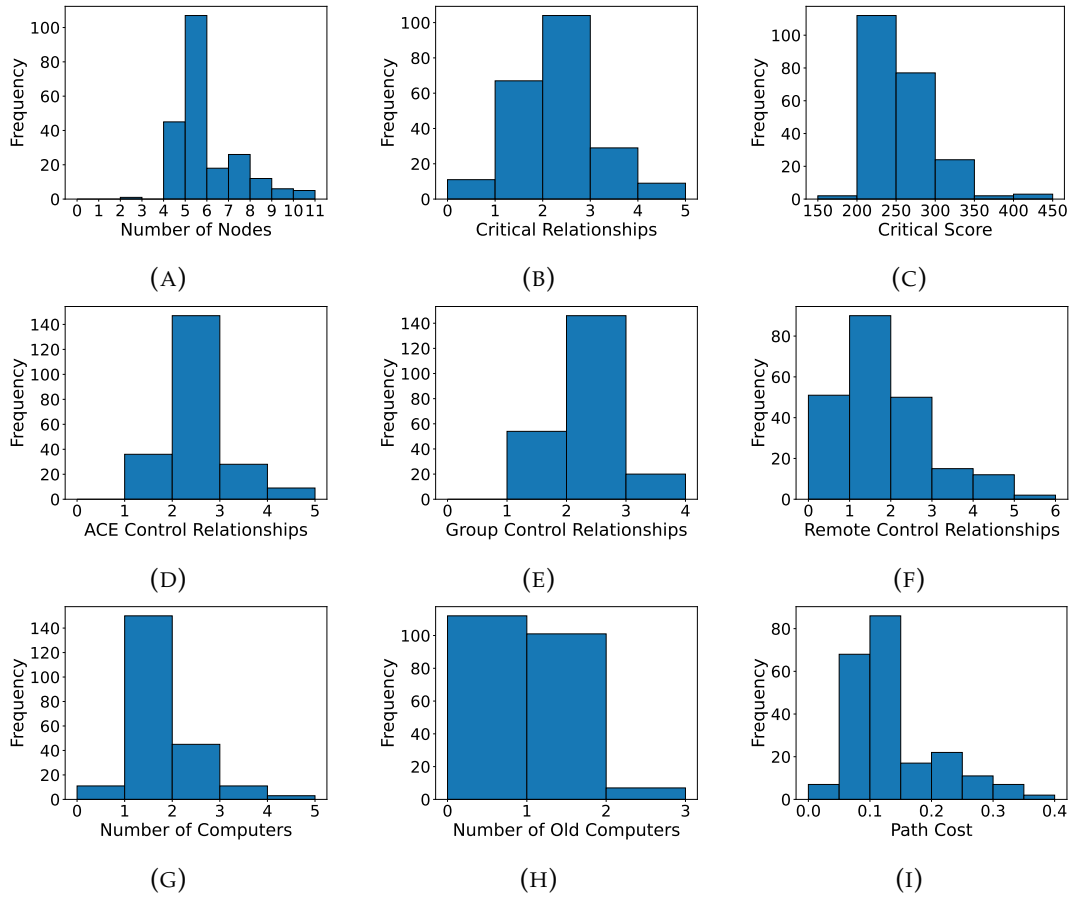


FIGURE 9.1: Distributions of the features related to weighted paths

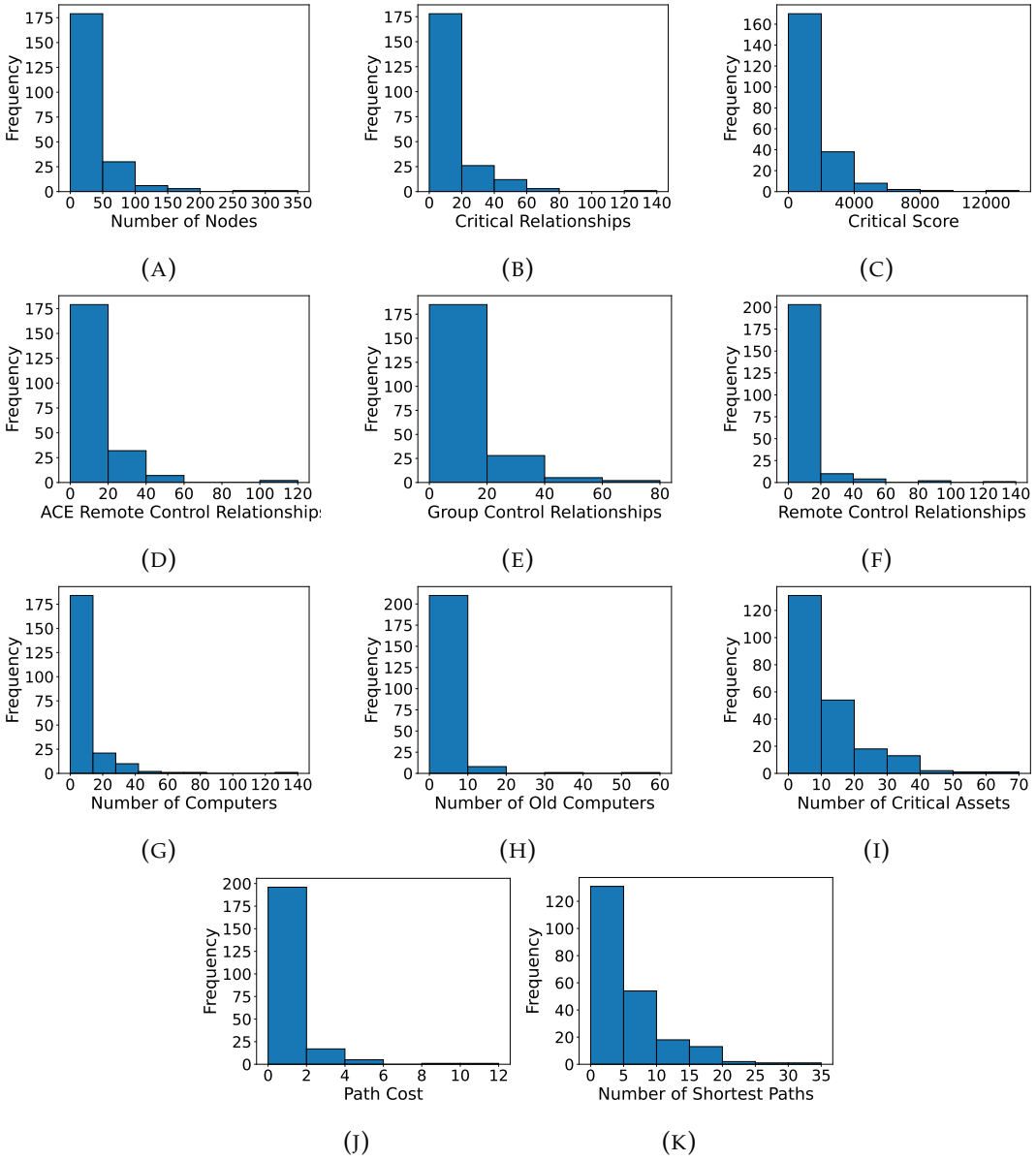


FIGURE 9.2: Distributions of the features related to non-weighted paths

From this dataset, the highly correlated features, that is, features with a high coefficient of correlation (i.e., above 0.9) have been removed. The remaining dataset is composed by 12 features.

Table 9.3 presents descriptive statistics of these features.

Feature	Mean	Std Dev	Min	Max	Median
Remote Relationships (weighted)	1.34	1.14	0	6	1
Group Nodes (weighted)	1.87	0.62	1	4	2
Critical Relationships (weighted)	1.81	0.89	0	5	2
ACE Relationships (weighted)	2.05	0.71	1	5	2
Path Cost (weighted)	0.13	0.069	0.01	0.37	0.11
Users (weighted)	2.33	0.61	1	5	2
Old Computers (weighted)	0.53	0.58	0	3	0
Computers (weighted)	1.29	0.70	0	4	1
Administrative Sessions (weighted)	0.92	0.27	0	1	1
Critical Score (weighted)	257.59	38.81	168	409	248
Path Cost	0.87	1.35	0.01	11.32	0.39
Old Computers	2.22	5.08	0	57	1

TABLE 9.3: Statistics of the features used for the classification process

The final dataset contains 220 data entries, corresponding to graph models described with the features listed in Table 9.3. Within this dataset, half of these models correspond to safe networks, while the other half corresponds to vulnerable networks.

Note that each shortest path has been labeled individually through a manual visual inspection of the sub-graphs according to the attacks described in Chapter 6. It is particularly important to perform manual labeling in these contexts and domain knowledge is an essential element to be able to determine whether a shortest path can lead to the compromise of the network.

9.2 Classification Performance

The classification task has been performed using different common classification algorithms, namely, logistic regression, support-vector machines and random forests. These algorithms generally represent good choices on rectangular data (i.e., structured information) and are relatively simple to tune with respect to other algorithms.

The classifiers have been tested with a k-fold validation technique, setting $k = 10$. This cross validation scheme has been chosen because of its benefits on small datasets.

An initial 80/20 split has been performed to use the 20% in the final evaluation of the classification system.

9.2.1 Hyper-Parameter Tuning

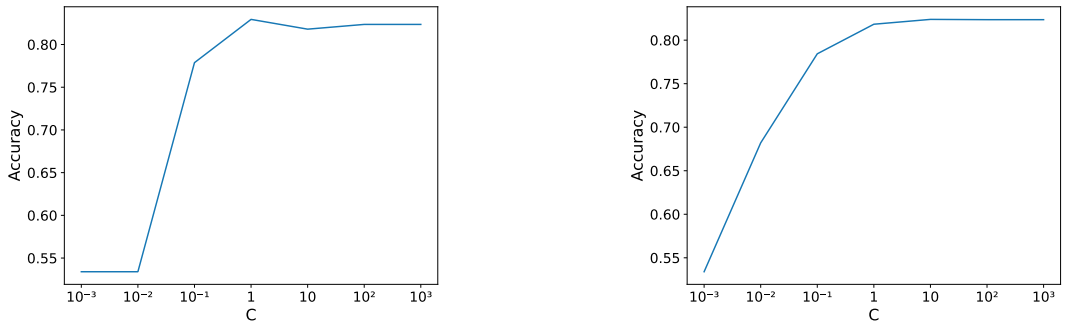
The tuning of hyper-parameters is based on a grid-search customized to each classifier.

Table 9.4 presents the hyper-parameters used for the grid search of the logistic regression (LR) classifier.

C	$10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}, 10^{3}$
Penalty	L1, L2

TABLE 9.4: Hyper-parameters used for tuning the Logistic Regression classifier

The accuracy corresponding to the tested hyper-parameters is shown in Figure 9.3.



(A) $L1$ Penalty (B) $L2$ Penalty

FIGURE 9.3: Accuracy of the Logistic Regression classifier on the Cross-Validation dataset

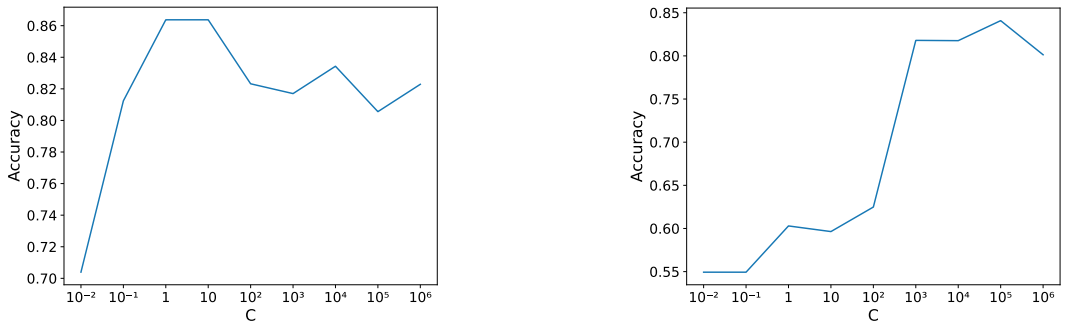
As can be seen, the results of the validation suggest that the best values of the regularization parameter C and of the penalty are equal to 1 and $L1$ respectively. In fact, the resulting accuracy on the testing dataset when using these hyper-parameters is equal to 84.1%.

The grid-search of the support-vector machine (SVM) classifier is based on the hyper-parameters presented in Table 9.5.

C	$10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5, 10^6$
Kernel	Linear, RBF

TABLE 9.5: Hyper-parameters used for tuning the Support Vector Machine classifier

The accuracy corresponding to the tested hyper-parameters is shown in Figure 9.4. The best performance using SVM during validation is obtained



(A) Using *Linear* Kernel (B) Using *RBF* Kernel

FIGURE 9.4: Accuracy of the Support-Vector Machine classifier on the Cross-Validation dataset

by setting the regularization parameter $C = 1$ and using a *linear* kernel. By using these hyper-parameters the resulting accuracy on the testing dataset is 86.3%.

For the Random Forest (RF), the grid search is based on two tuning parameters, namely, the number of decision trees used within the forest and maximum tree depth.

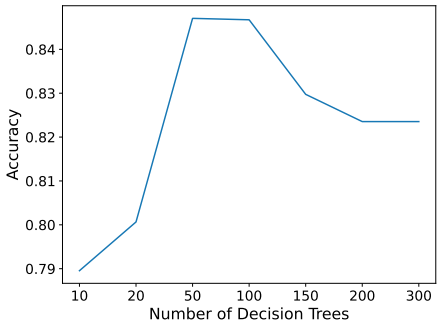
Table 9.6 presents the details of these parameters. The accuracy corresponding to the tested hyper-parameters is shown in Figure 9.5.

Number of Decision Trees	10, 20, 50, 100, 150, 200, 300
Max Depth for Trees	None, 2, 3, 4

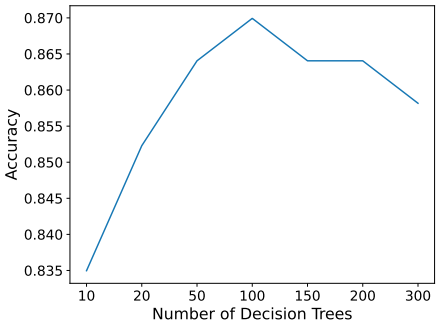
TABLE 9.6: Hyper-parameters used for tuning the Random Forest classifier

As can be seen, the grid search suggests that the best performance is obtained with a number of decision trees equal to 100 and a max depth for trees equal to 3. By using these best hyper-parameters the resulting accuracy reaches 91% on the testing dataset.

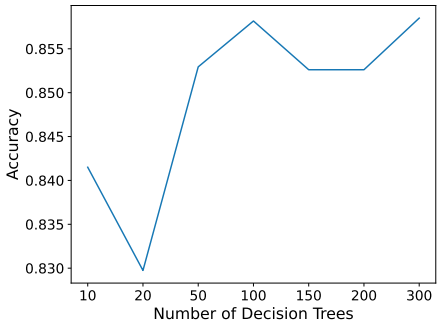
Another interesting result is related to the importance of features. When using Random Forests, it is possible to determine the importance of each feature for the classification task. Figure 9.6 shows a bar plot that highlights the most important features involved in the classification process.



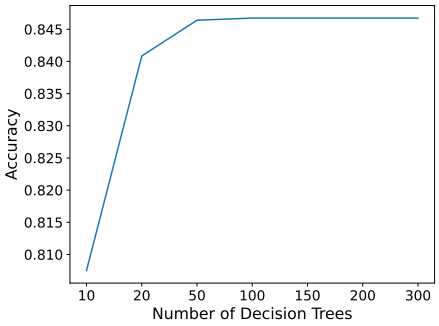
(A) $MaxDepth = 2$



(B) $MaxDepth = 3$



(C) $MaxDepth = 4$



(D) *No limit on depth*

FIGURE 9.5: Accuracy of the Random Forest classifier on the Cross-Validation dataset

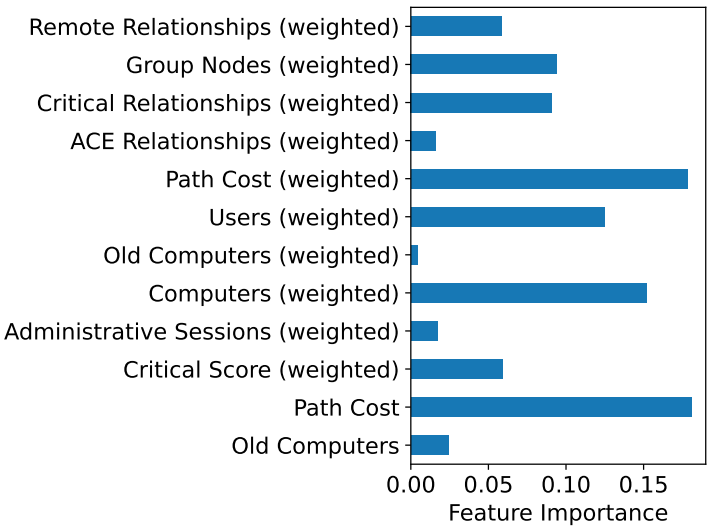


FIGURE 9.6: Feature importance for the Random Forest classifier

These results are particularly interesting, since they show how the definition of critical scores and path costs are fundamental to obtain good classification results. In fact, some of the most important features correspond to characteristics of the weighted shortest path.

9.2.2 Performance Comparison

The performance of the three classifiers, with respect to the security assessment of the Active Directory environments, is summarized in Table 9.7. Note that this performance, expressed in terms of precision, recall and F1-score, refers to the testing dataset.

Classifier	Label	Precision	Recall	F1
LR	Safe	0.74	0.88	0.80
	Vulnerable	0.92	0.82	0.87
SVM	Safe	0.90	0.82	0.86
	Vulnerable	0.83	0.91	0.87
RF	Safe	0.95	0.86	0.90
	Vulnerable	0.88	0.95	0.91

TABLE 9.7: Comparison of the performance of the three classifiers

In general, the performance of all the classifiers evaluated is good. For example, the F1-score ranges between 0.80 obtained by the LR classifier for the assessment of safe networks and 0.91 obtained by the RF classifier for the assessment of vulnerable networks. Let us remark that random forests are in practice well known to often outperform other classifiers in many applications. In our specific case, about nine times out of ten, random forests are able to correctly detect whether a complex network environment is vulnerable or not.

The F1-scores reflect the accuracy results obtained on the testing dataset for all of the analyzed classifiers. This is due to the balanced nature of the dataset on which the classifiers have been trained, validated and tested.

In conclusion, these results validate the benefits of using a classification system in performing automated network assessment.

Chapter 10

Conclusion

Penetration testing is an important activity for evaluating the security of technological infrastructures and services. Security assessment is becoming a fundamental and necessary step within the systems and software development life-cycle. This thesis work focused on the design and development of a methodology and framework for performing AI-assisted penetration testing activities within enterprise environments. The proposed methodology relies on graph theory and machine learning techniques coupled with solid domain knowledge.

The development of the framework involved the design of a methodological framework and of a toolset for network assessment based on AI. The toolset is designed in a modular structure and each tool has a specific purpose. The entire toolchain includes the enumeration phase and information gathering process as well as the detection of vulnerabilities within a complex infrastructure.

A fundamental part of the framework is the approach proposed to assign scores to nodes and costs to edges, thus enabling the identification of weighted shortest paths based on the Djkstra's algorithm. The experimental results have shown that the proposed solutions allow an accurate identification of the most critical entities in an enterprise network.

In particular, ensemble methods, e.g., Random Forests, coupled with feature engineering obtain good performance in the detection of vulnerable networks. The accuracy on the test dataset is about 91%.

The entire toolset accompanying this framework, that is, the automated enumeration tools, the threat modeling tool, the simulator of Active Directory domains and the classification system will be released as open-source software.

As a future work there are still many open research issues to be explored. For example, more complex attack scenarios could be investigated, such as advanced delegation attacks or diversification of privilege escalation techniques. Moreover, models of additional lateral movement techniques when no shortest path is available for a node should be investigated. More precisely, the possibility of compromising other users starting from a user with no paths available.

Another possible research direction refers to the refinement of the graph generator to allow testing forests containing multiple domains and the generation of even more realistic environments.

Bibliography

[1] Guido Grillenmeier. Now's the time to rethink Active Directory security. *Network Security*, 2021(7):13–16, 2021.

[2] William Stallings, Lawrie Brown, Michael D. Bauer, and Arup Kumar Bhattacharjee. *Computer security: principles and practice*. Pearson, 2012.

[3] Stephen McCombie et al. Threat actor oriented strategy: Knowing your enemy to better defend, detect and respond to cyber-attacks. *Journal of the Australian Institute of Professional Intelligence Officers*, 26(1):24, 2018.

[4] Adrien Bonguet and Martine Bellaiche. A survey of denial-of-service and distributed denial of service attacks and defenses in cloud computing. *Future Internet*, 9(3):43, 2017.

[5] Aparna Verma, M.S. Rao, A.K. Gupta, W. Jeberson, and Vrijendra Singh. A literature review on malware and its analysis. *International Journal of Current Research and Review*, 5(16):71, 2013.

[6] Kang Leng Chiew, Kelvin Sheng Chek Yong, and Choon Lin Tan. A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications*, 106:1–20, 2018.

[7] Long Cheng, Fang Liu, and Danfeng Yao. Enterprise data breach: causes, challenges, prevention, and future directions. *Data Mining and Knowledge Discovery*, 7(5):e1211, 2017.

[8] Eric Cole. *Advanced persistent threat: understanding the danger and how to protect your organization*. Newnes, 2012.

[9] Omar G. Abood and Shawkat K Guirguis. A survey on cryptography algorithms. *International Journal of Scientific and Research Publications*, 8(7):410–415, 2018.

[10] Byoungjin Seok, Jose Costa Sapalo Sicato, Tcydenova Erzhena, Canshou Xuan, Yi Pan, and Jong Hyuk Park. Secure D2D communication for 5G IoT network based on lightweight cryptography. *Applied Sciences*, 10(1):217, 2020.

[11] Phap Duong-Ngoc, Tuy Nguyen Tan, and Hanho Lee. Efficient newhope cryptography based facial security system on a GPU. *IEEE Access*, 8:108158–108168, 2020.

[12] Nilupulee A. Gunathilake, William J. Buchanan, and Rameez Asif. Next generation lightweight cryptography for smart iot devices:: implementation, challenges and applications. In *Proc. of 2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 707–710. IEEE.

[13] Sagarika Ghosh and Srinivas Sampalli. A survey of security in SCADA networks: Current issues and future challenges. *IEEE Access*, 7:135812–135831, 2019.

[14] Masoumeh Sharafi, Faranak Fotouhi-Ghazvini, Mohsen Shirali, and Mona Ghassemian. A low power cryptography solution based on chaos theory in wireless sensor nodes. *IEEE Access*, 7:8737–8753, 2019.

[15] Fabio Campos, Lars Jellema, Mauk Lemmen, Lars Müller, Daan Sprenkels, and Benoit Viguier. Assembly or optimized C for lightweight cryptography on RISC-V? In *Proc. International Conference on Cryptology and Network Security*, pages 526–545. Springer, 2020.

[16] Dominic Mayers. Unconditional security in quantum cryptography. *Journal of the ACM*, 48(3):351–406, 2001.

[17] Stefano Pirandola, Ulrik L. Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, Dirk Englund, Tobias Gehring, Cosmo Lupo, Carlo Ottaviani, et al. Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4):1012–1236, 2020.

[18] Sergei Nikolaevich Molotkov. On the secrecy of a simple and effective implementation of BB84 quantum cryptography protocol. *Laser Physics Letters*, 16(7), 2019.

[19] Juan Yin, Yu-Huai Li, Sheng-Kai Liao, Meng Yang, Yuan Cao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Shuang-Lin Li, et al. Entanglement-based secure quantum cryptography over 1,120 kilometres. *Nature*, 582(7813):501–505, 2020.

[20] Derek Hall and Timothy Sands. Quantum cryptography for nuclear command and control. *Computer and Information Science*, 13(1):1–72, 2020.

[21] Wenjun Xiong and Robert Lagerström. Threat modeling–a systematic literature review. *Computers & Security*, 84:53–69, 2019.

[22] Abel Yeboah-Ofori and Shareeful Islam. Cyber security threat modeling for supply chain organizational environments. *Future Internet*, 11(3):63, 2019.

[23] Kim Wuyts, Dimitri Van Landuyt, Aram Hovsepyan, and Wouter Joosen. Effective and efficient privacy threat modeling through domain refinements. In *Proc. of the 33rd Annual ACM Symposium on Applied Computing*, pages 1175–1178, 2018.

[24] Dimitri Van Landuyt, Laurens Sion, Emiel Vandeloo, and Wouter Joosen. On the applicability of security and privacy threat modeling for blockchain applications. In S. Katsikas et al., editors, *Computer Security*, Lecture Notes in Computer Science, pages 195–203. Springer, 2019.

[25] Aleksey Novokhrestov, Anton Konev, Alexander Shelupanov, and Alexander Buymov. Computer network threat modelling. *Journal of Physics: Conference Series*, 1488, 2020.

[26] Laurens Sion, Koen Yskout, Dimitri Van Landuyt, Alexander van den Berghe, and Wouter Joosen. Security threat modeling: Are data flow diagrams enough? In *Proc. of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 254–257, 2020.

[27] Franco Loi, Arunan Sivanathan, Hassan Habibi Gharakheili, Adam Radford, and Vijay Sivaraman. Systematically Evaluating Security and Privacy for Consumer IoT Devices. In *Proc. of the Workshop on Internet of Things Security and Privacy (IoTS&P)*. ACM, 2017.

[28] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. SoK: Security Evaluation of Home-Based IoT Deployments. In *Proc. of the IEEE Symposium on Security and Privacy (S&P)*, pages 1362–1380, 2019.

[29] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys & Tutorials*, 21(3):2702–2733, 2019.

[30] Mohab Aly, Foutse Khomh, Mohamed Haoues, Alejandro Quintero, and Soumaya Yacout. Enforcing security in Internet of Things frameworks: A Systematic Literature Review. *Internet of Things*, 6:100050, 2019.

[31] L. Mary Shamala, Godandapani Zayaraz, K. Vivekanandan, and V. Vijayalakshmi. Lightweight cryptography algorithms for internet of things enabled networks: An overview. *Journal of Physics: Conference Series*, 1717:012072, 2021.

[32] Wei Zhou, Yan Jia, Anni Peng, Yuqing Zhang, and Peng Liu. The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved. *IEEE Internet of Things Journal*, 6(2):1606–1616, 2019.

[33] Minhaj Ahmad Khan. A survey of security issues for cloud computing. *Journal of Network and Computer Applications*, 71:11–29, 2016.

[34] Hamed Tabrizchi and Marjan Kuchaki Rafsanjani. A survey on security challenges in cloud computing: issues, threats, and solutions. *The Journal of Supercomputing*, 76(12):9493–9532, 2020.

[35] Rakesh Kumar and Rinkaj Goyal. On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. *Computer Science Review*, 33:1–48, 2019.

[36] Reddy SaiSindhuTheja and Gopal K. Shyam. An efficient metaheuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud computing environment. *Applied Soft Computing*, 100, 2020.

[37] Juan Camilo Correa Chica, Jenny Cuatindioy Imbachi, and Juan Felipe Botero Vega. Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, 159, 2020.

[38] Hongxin Hu, Wonkyu Han, Gail-Joon Ahn, and Ziming Zhao. Flowguard: building robust firewalls for software-defined networks. In *Proc. of the Third Workshop on Hot topics in Software Defined Networking*, pages 97–102, 2014.

[39] Marcos VO de Assis, Luiz F Carvalho, Joel JPC Rodrigues, Jaime Lloret, and Mario L Proença Jr. Near real-time security system applied to SDN environments in IoT networks using convolutional neural network. *Computers & Electrical Engineering*, 86, 2020.

[40] Tushaar Gangavarapu, C.D. Jaidhar, and Bhabesh Chanduka. Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artificial Intelligence Review*, pages 1–63, 2020.

[41] Sahar Bosaeed, Iyad Katib, and Rashid Mehmood. A Fog-Augmented Machine Learning based SMS Spam Detection and Classification System. In *Proc. of the 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 325–330. IEEE, 2020.

[42] Tazmina Sharmin, Fabio Di Troia, Katerina Potika, and Mark Stamp. Convolutional Neural Networks for image spam detection. *Information Security Journal: A Global Perspective*, 29(3):103–117, 2020.

[43] S. Sibi Chakkaravarthy, Dhamodara Sangeetha, and V. Vaidehi. A survey on malware analysis and mitigation techniques. *Computer Science Review*, 32:1–23, 2019.

[44] Seong Il Bae, Gyu Bin Lee, and Eul Gyu Im. Ransomware detection using machine learning algorithms. *Concurrency and Computation: Practice and Experience*, 32(18):e5422, 2020.

[45] Nelson Ochieng, Waweru Mwangi, and Ismail Ateya. Generalization Performance Comparison of Machine Learners for the Detection of Computer Worms Using Behavioral Features. In Millie Pant, Kumar T. Sharma, Arya Rajeev, Bikash Sahana, and Hossein Zolfagharinia, editors, *Soft Computing: Theories and Applications*, volume 1154 of *Advances in Intelligent Systems and Computing*, pages 677–693. Springer, 2020.

[46] Ning Shang, An Wang, Yaoling Ding, Keke Gai, Liehuang Zhu, and Guoshuang Zhang. A machine learning based golden-free detection method for command-activated hardware trojan. *Information Sciences*, 540:292–307, 2020.

[47] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147, 2019.

[48] Moutaz Alazab, Mamoun Alazab, Andrii Shalaginov, Abdelwadood Mesleh, and Albara Awajan. Intelligent mobile malware detection using permission requests and API calls. *Future Generation Computer Systems*, 107:509–521, 2020.

[49] Jueun Jeon, Seungyeon Baek, Minho Kim, Inho Go, and Young-Sik Jeong. IoT Malware Dynamic Analysis Scheme Using the CNN Model. In James J. Park, Simon James Fong, Yi Pan, and Yunsick Sung, editors, *Advances in Computer Science and Ubiquitous Computing*, volume 715 of *Lecture Notes in Electrical Engineering*, pages 547–553. Springer, 2021.

[50] Yuxin Ding, Xiao Zhang, Jieke Hu, and Wenting Xu. Android malware detection method based on bytecode image. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10, 2020.

[51] Andrew McDole, Mahmoud Abdelsalam, Maanak Gupta, and Sudip Mittal. Analyzing CNN based behavioural malware detection techniques on cloud iaas. In *Proc. of the International Conference on Cloud Computing*, pages 64–79. Springer, 2020.

[52] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138, 2020.

[53] Sunanda Gamage and Jagath Samarabandu. Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767, 2020.

[54] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124, 2020.

[55] Sydney Mambwe Kasongo and Yanxia Sun. A deep learning method with filter based feature engineering for wireless intrusion detection system. *IEEE Access*, 7:38597–38607, 2019.

[56] Muder Almiani, Alia AbuGhazleh, Amer Al-Rahayfeh, Saleh Atiewi, and Abdul Razaque. Deep recurrent neural network for IoT intrusion detection system. *Simulation Modelling Practice and Theory*, 101:102031, 2020.

[57] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *Proc. of the 1998 Workshop on New security paradigms*, pages 71–79. ACM Press, 1998.

[58] Ronald W. Ritchey and Paul Ammann. Using model checking to analyze network vulnerabilities. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 156–165. IEEE, 2000.

[59] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002.

[60] Somesh Jha, Oleg Sheyner, and Jeannette Wing. Two formal analyses of attack graphs. In *Proc. of the 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pages 49–63. IEEE, 2002.

[61] John Dunagan, Alice X. Zheng, and Daniel R. Simon. Heat-ray: combating identity snowball attacks using machine learning, combinatorial optimization and attack graphs. In *Proc. of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, pages 305–320, 2009.

[62] Nwokedi Idika and Bharat Bhargava. Extending attack graph-based security metrics and aggregating their application. *IEEE Transactions on dependable and secure computing*, 9(1):75–85, 2010.

[63] Jorge Lucangeli Obes, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. *arXiv preprint arXiv:1306.4044*, 2013.

[64] Nirnay Ghosh, Ishan Chokshi, Mithun Sarkar, Soumya K. Ghosh, Anil K. Kaushik, and Sajal K. Das. NetSecuritas: An integrated Attack Graph-based Security Assessment Tool for Enterprise Networks. In *Proc. of the International Conference on Distributed Computing and Networking*, 2015.

[65] Valentina Casola, Alessandra De Benedictis, Massimiliano Rak, and Umberto Villano. A methodology for automated penetration testing of cloud applications. *International Journal of Grid and Utility Computing*, 11(2):267–277, 2020.

[66] Jiayin Wang, Teng Wang, Zhengyu Yang, Ying Mao, Ningfang Mi, and Bo Sheng. SEINA: A stealthy and effective internal attack in Hadoop systems. In *Proc. of the International Conference on Computing, Networking and Communications ICNC*, pages 525–530. IEEE, 2017.

[67] Gemini George and Sabu M Thampi. A graph-based security framework for securing industrial IoT networks from vulnerability exploitations. *IEEE Access*, 6:43586–43601, 2018.

[68] Vasaka Visoottiviseth, Phuripat Akarasiriwong, Siravitch Chaiyasart, and Siravit Chotivatunyu. Pentos: Penetration testing tool for Internet

of Thing devices. In *Proc. of TENCON IEEE Region 10 Conference*, pages 2279–2284, 2017.

[69] Georgios Apostolopoulos. Graph-Based Network Security Threat Detection Across Time and Entities. US Patent 10,205,735, 2019.

[70] Kevin Roundy, Fanglu Guo, Sandeep Bhatkar, Tao Cheng, Jie Fu, Zhi Kai Li, Darren Shou, Sanjay Sawhney, Acar Tamersoy, Elias Khalil, et al. Systems and methods for using event-correlation graphs to detect attacks on computing systems. US Patent 9,141,790, 2015.

[71] Martin Vejman and Lukas Machlica. Graph prioritization for improving precision of threat propagation algorithms. US Patent 10,523,691, 2019.

[72] Michael Jon Swan. Discovery and visualization of Active Directory domain controllers in topological network maps. US Patent 8,045,486, 2011.

[73] Sudhakar Muddu and Christos Tryfonas. Detection of clustering in graphs in network security analysis. US Patent 10,003,605, 2018.

[74] Elvira Ismagilova, Laurie Hughes, Nripendra P. Rana, and Yogesh K. Dwivedi. Security, privacy and risks within smart cities: Literature review and development of a smart city interaction framework. *Information Systems Frontiers*, 2020.

[75] Imane Horiya Brahmi, Nirwan Ansari, Mubashir Husain Rehmani, et al. Cyber security framework for vehicular network based on a hierarchical game. *IEEE Transactions on Emerging Topics in Computing*, 9(1):429–440, 2019.

[76] Zesheng Xi, Jie Cui, and Bo Zhang. Research on automated penetration testing framework for power web system integrating property information and expert experience. In *Proc. of the 9th International Conference on Computer Engineering and Networks*, pages 855–865. Springer, 2021.

[77] Prabhakar Krishnan, Subhasri Duttagupta, and Krishnashree Achuthan. Varman: Multi-plane security framework for software defined networks. *Computer Communications*, 148:215–239, 2019.

[78] Ilias Giechaskiel, Youqian Zhang, and Kasper B. Rasmussen. A framework for evaluating security in the presence of signal injection attacks. In *European Symposium on Research in Computer Security*, pages 512–532. Springer, 2019.

[79] Jalal Bhayo, Sufian Hameed, and Syed Attique Shah. An efficient counter-based DDoS attack detection framework leveraging software defined IoT (SD-IoT). *IEEE Access*, 8:221612–221631, 2020.

[80] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 5GReasoner: A property-directed security and privacy analysis framework for 5G cellular network protocol. In *Proc. of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 669–684, 2019.

[81] Wu Xindong. Inductive learning. *Journal of Computer Science and Technology*, 8(2):118, 1993.

[82] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2):206–226, 2000.

[83] Tom M. Mitchell. *Machine learning*. McGraw Hill, 1997.

[84] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[85] Stuart Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson, 2010.

[86] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly, 2017.

[87] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[88] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[89] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT Press, 2012.

[90] Kevin P. Murphy. *Machine Learning: A probabilistic perspective*. MIT Press, 2012.

[91] Peter M. Lee. *Bayesian Statistics: An introduction*. John Wiley & Sons, 2012.

[92] Andrew Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. In *Conference on Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[93] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[94] Pavel Pudil and Jana Novovičová. Novel methods for feature subset selection with respect to problem knowledge. In Huan Huan LiuHiroshi and Hiroshi Motoda, editors, *Feature Extraction, Construction and Selection*, volume 453 of *Series in Engineering and Computer Science*, pages 101–116. Springer, 1998.

[95] Gregory Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. *Knowledge Discovery in Databases*, pages 229–248, 1991.

[96] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[97] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009.

[98] Claude Sammut and Geoffrey I. Webb. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[99] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, 1997.

[100] Ethem Alpaydin. *Introduction to machine learning*. MIT Press, 2014.

[101] Raymond E Wright. Logistic regression. 1995.

[102] Ms Hetal Bhavsar and Amit Ganatra. Radial basis polynomial kernel (rbpk): A generalized kernel for support vector machine. *International Journal of Computer Science and Information Security (IJCSIS)*, 14(4), 2016.

[103] Hetal Bhavsar and Amit Ganatra. Variations of support vector machine classification technique: a survey. *International Journal of Advanced Computer Research*, 2(6):230–236, 2012.

[104] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[105] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC Press, 1984.

[106] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(4):476–487, 2005.

[107] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[108] David W Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[109] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[110] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39, 2010.

[111] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[112] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1):85–103, 1999.

[113] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[114] Tin Kam Ho. Random decision forests. In *Proc. of the Third International Conference on Document Analysis and Recognition*, volume 1, pages 278–282, 1995.

[115] Douglas Brent West. *Introduction to graph theory*, volume 2. Prentice Hall, 2001.

[116] Reinhard Diestel. Graph theory. *Graduate texts in mathematics*, 173, 2005.

[117] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017.

[118] Kairanbay Magzhan and Hajar Mat Jani. A review and evaluations of shortest path algorithms. *International Journal of Scientific & Technology Research*, 2(6):99–104, 2013.

[119] Narsingh Deo and Chi-Yin Pang. Shortest-path algorithms: Taxonomy and annotation. *Networks*, 14(2):275–323, 1984.

[120] Boris V Cherkassky, Andrew V Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical programming*, 73(2):129–174, 1996.

[121] Paul Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, 1987.

[122] Paul Mockapetris. Domain Names - Implementation and Specification. RFC 1035, 1987.

[123] Arnt Gulbrandsen and Paul Vixie. A DNS RR for specifying the location of services (DNS SRV). RFC 2052, 1996.

[124] Paul Vixie, Susan Thomson, Yakov Rekhter, and Jim Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, 1997.

[125] J. Sermersheim. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511, 2006.

[126] K. Zeilengal. Lightweight Directory Access Protocol (LDAP): Directory Information Models. RFC 4512, 2006.

[127] R. R. Harrison. Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms. RFC 4513, 2006.

[128] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names. RFC 4514, 6 2006.

[129] M. Smith and T. Howes. Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters. RFC 4515, 2006.

[130] M. Smith and T. Howes. Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator. RFC 4516, 2006.

[131] S. Legg. Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules. RFC 4517, 2006.

[132] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation. RFC 4518, 2006.

[133] A. Sciberras. Lightweight Directory Access Protocol (LDAP): Schema for User Applications. RFC 4519, 2006.

[134] A. Sciberras. The LDAP Data Interchange Format (LDIF) - Technical Specification. RFC 2849, 2000.

[135] S. Legg. Lightweight Directory Access Protocol (LDAP): The Binary Encoding Option. RFC 4522, 2006.

[136] Alexey Melnikov and Kurt Zeilenga. Simple Authentication and Security Layer (SASL). RFC 4422, 2006.

[137] Kurt Zeilenga. Anonymous Simple Authentication and Security Layer (SASL) Mechanism. RFC 4505, 2006.

[138] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, 2005.

[139] J. Brezak M. Swift, J. Trostle. Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols. RFC 3244, 2002.

[140] Protocol Standard for a NetBIOS service on a TCP/UDP transport: concepts and methods. RFC 1001, 1987.

[141] Protocol Standard for a NetBIOS service on a TCP/UDP transport: detailed specifications. Technical Report 1002, 1987.

[142] Sun Microsystems Inc. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 1057, 1988.

[143] Robert Thurlow. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 5531, 2009.

[144] Microsoft Corporation. *Remote Procedure Call Protocol Extensions*. Microsoft Corporation, 2018.

[145] Microsoft Corporation. *Understanding the Remote Desktop Protocol (RDP)*, 2017.

[146] Microsoft Corporation. *NT LAN Manager (NTLM) Authentication Protocol*. Microsoft Corporation, 2020.

[147] Ronald Rivest. The MD5 Message-Digest Algorithm. RFC 1321, 1992.

[148] Ronald Rivest. The MD4 Message-Digest Algorithm. RFC 1320, 1992.

[149] Microsoft Corporation. *NTLM: Security Considerations for Implementers*, 2020.

[150] Bernard Aboba, David Thaler, and Levon Esibov. Link-Local Multicast Name Resolution (LLMNR). RFC 4795, 2007.

[151] Joao Damas, Michael Graff, and Paul Vixie. Extension Mechanisms for DNS (EDNS0). RFC 6891, 2013.

[152] Paul Vixie. Extension Mechanisms for DNS (EDNS0). RFC 2671, 1999.

[153] Microsoft Corporation. *Web Services Management Protocol Extensions for Windows Server 2003*. Microsoft Corporation, 2017.

[154] Distributed Management Task Force Inc. Web Services for Management (WS-Management) Specification. Technical Report DSP0226, 2008.

[155] Maciej Besta, Emanuel Peter, Robert Gerstenberger, Marc Fischer, Michał Podstawski, Claude Barthels, Gustavo Alonso, and Torsten Hoefler. Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. *arXiv preprint arXiv:1910.09017*, 2019.

[156] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

[157] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer, 2013.