UNIVERSITÀ DEGLI STUDI DI SALERNO

Department of Information and Electrical Engineering and applied Mathematics

Master's Degree in Computer Engineering

Cybersecurity Curriculum

DESIGN AND APPLICATION OF A SECURITY ASSESSMENT AND HARDENING METHODOLOGY FOR VIRTUAL ENVIRONMENTS

First Supervisor
Prof. Vincenzo Carletti
Second Supervisor
Prof. Pasquale Foggia

Candidate
Enrico Maria Di Mauro
Identification Number
0622701706

Academic Year 2022/2023

Table of contents

Chapter 1. Introduction

1.1. Problem definition

In the digital era, data and information security are becoming increasingly crucial. Companies and organizations of all sizes are constantly exposed to cyber threats that can have a significant impact on their operations.

Since all systems can be vulnerable to cyberattacks, it is important to conduct security assessments and implement appropriate hardening measures. A security assessment is a process that identifies and evaluates vulnerabilities in a computer system. The goal of this operation is to understand the risk posed by potential cyberattacks.

Hardening, on the other hand, is a process that enhances the security of a computer system, with the aim of reducing the possibility of attacks.

In recent years, there has been a significant increase in cyber threats, attributable to various factors, including:

- The growing complexity of computer systems
- The proliferation of cloud technologies
- Increased demand for sensitive data

Cyber threats can have a significant impact on the activities of businesses and organizations, including data loss, privacy breaches, system disruptions, and financial theft.

In many cases, companies and organizations may not pay sufficient attention to information security, often due to factors such as:

- Lack of resources
- Lack of expertise
- Lack of awareness

Superficial security measures can make companies and organizations more vulnerable to cyberattacks.

Based on this information, after understanding the importance of keeping a computer system secure, a process was designed as part of the thesis project to conduct security assessments and subsequent system hardening. The latter was then applied and validated on virtual machines and the virtualization platform used, namely Proxmox, provided by the DIEM (Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica applicata) at the University of Salerno. The guidelines provided by the Open Source Security Testing

Methodology Manual (OSSTMM) were found useful for these operations, allowing for reliable and repeatable results and ensuring that the system is protected from potential threats and vulnerabilities.

1.2. Relevance of the problem in computer engineering context

During the development of the thesis, an engineering approach was adopted as the methodological foundation to systematically design the entire process of system analysis and security. This approach was instrumental in ensuring a thorough design with the objective of creating a generalized model, one that is adaptable and applicable not only to the specific machines under examination but also to potential future contexts.

The engineering approach was structured into various phases, each of which had a clear justification for its relevance within the overall project. Initially, a planning phase was initiated in which the research objectives were defined. This allowed for the precise identification of which systems would be the subject of analysis.

A crucial phase involved the analysis and evaluation of the systems, following the guidelines provided by OSSTMM. This approach enabled a detailed and comprehensive analysis aimed at identifying vulnerabilities, risks, and potential threats. The goal was to produce an in-depth report that provided a clear understanding of the security issues related to the systems under examination.

A key element of the process was the assessment of the attack surface of the machines. This was accomplished through the use of Risk Assessment Values (RAVs), which identified critical points in the systems and provided a detailed awareness of possible entry points for external or internal attacks.

A distinctive aspect of the approach was its iterative nature. This constant iteration allowed for a continuous cycle of assessment and improvement. After implementing specific security measures, activities were repeated to assess the effectiveness of the adopted solutions and to identify new vulnerabilities or potential changes in the security context. This method played a crucial role in maintaining a consistently high level of system security, adapting it to evolving threats and technological developments.

In the design of the activities, significant emphasis was placed on replicability and consistency. This was done with the aim of ensuring that the repeated application of the methodology, without a change in context, would yield consistent results in line with the

initial findings. Furthermore, the methodologies and solutions were developed to be applicable successfully in any context. Creating a generalized process allowed for reliable and comparable results, representing a significant advantage for the validity and applicability of the approach itself.

Additionally, it is essential to understand the crucial importance of safeguarding sensitive data in the context of system analysis and security. The careful preservation and protection of sensitive data are fundamental aspects in the design and implementation of a robust engineering approach.

The loss or compromise of sensitive data can lead to a range of significant consequences, varying based on the type of information involved. In framing this concept, the following potential implications can be outlined:

- The exposure of Personally Identifiable Information (PII) risks triggering identity theft and infringing upon individuals' privacy, leading to devastating consequences such as financial fraud and reputational damage. Financial data is equally sensitive, and its unintended disclosure can result in financial fraud and scams causing economic and legal disruptions for victims.
- Intellectual property constitutes a fundamental component of innovation and corporate advancement. Its compromise can lead to substantial economic damages and competitive losses, allowing competitors to copy strategies and innovations effortlessly.
- The loss or unauthorized access to medical and healthcare data can unleash a series of challenges, including breaches of medical confidentiality and impacts on patient safety. Sensitive corporate data, if revealed, can jeopardize competitive advantage and an organization's reputation, causing financial impacts and damage to its public image.
- Considering broader perspectives, the disclosure of government and sensitive data can jeopardize national security, while the compromise of customer data can erode consumer trust, leading to financial losses and reputational damage. Additionally, the loss of sensitive data can result in regulatory compliance violations, subject to sanctions and legal liabilities.

It is evident that the security of sensitive data plays a fundamental role in today's ecosystem, influencing individuals' privacy, organizations' financial stability, and overall trust in the use of digital technologies. A well-structured engineering approach not only

ensures data protection but also contributes to establishing an environment where information is safeguarded with the utmost care and attention.

Chapter 2. State of the art

2.1. Detailed analysis of the state of the art

In the academic and corporate context, the need to efficiently and scalably manage a variety of computer systems has led to the widespread use of virtual machines (VMs). VMs represent a fundamental solution for overcoming the limitation of hardware resources, enabling the execution of different isolated operating environments within a single physical machine. This is particularly advantageous in situations where resource demands can vary widely over time.

The creation and management of VMs are made possible by hypervisors, which act as intermediaries between the physical hardware and the virtual machines. Hypervisors are essential for efficiently allocating and sharing system resources among different VMs, ensuring the necessary isolation and security for VMs to operate without interference. There are two main types of hypervisors: Type 1, known as Bare Metal, and Type 2, known as Hosted.

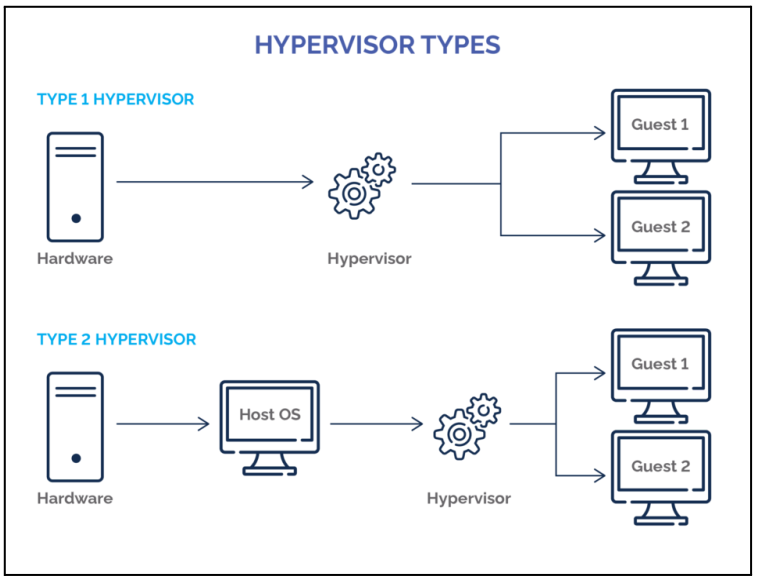

Figure 1. Hypervisor types

Type 1 hypervisors are dedicated operating systems that have direct access to the underlying hardware. This type of hypervisor offers high performance to VMs because they operate directly on it, without the mediation of a guest operating system. Type 1

hypervisors are preferred in scenarios where maximizing performance and ensuring VM isolation are crucial.

Type 2 hypervisors, on the other hand, are applications running on a host operating system. Although they may introduce a slight performance penalty due to the mediation of the host operating system, they offer greater flexibility. These hypervisors can be easily installed on existing systems without requiring dedicated infrastructure. The choice between Type 1 and Type 2 hypervisors depends on the specific needs of the environment and the priorities of performance and flexibility [1].

The adoption of VMs and hypervisors introduces new challenges related to cybersecurity. Hackers may target hypervisors to compromise the entire virtual environment. Two common attacks that jeopardize the security of hypervisors and VMs are VM Escape and Hyper Jacking.

VM Escape is an attack in which an attacker attempts to break the isolation between VMs and the host, gaining access to the host and subsequently to other VMs. Once the host is compromised, the attacker can damage the VMs or even take control of the entire physical system. This type of attack questions the robustness of the isolation provided by hypervisors.

Hyper Jacking, on the other hand, is an attack that aims to gain direct control of the hypervisor. If an attacker succeeds in compromising the hypervisor itself, they could jeopardize the entire virtual environment and the VMs within it. This attack underscores the critical importance of protecting the hypervisor from external threats.

Furthermore, another aspect that requires attention is VM Sprawl, which represents the excessive proliferation of VMs within a virtual environment. This can lead to poor resource management, causing availability and security issues. If VMs are not adequately monitored and managed, unused or overlapping resources may be allocated, leading to inefficiencies and potential vulnerabilities.

In addition to attacks that jeopardize the security of hypervisors and VMs, there are other threats that can directly target the VMs themselves. These attacks can have significant consequences on the security and integrity of virtualized resources within the environment used.

One of the most well-known attacks is Denial of Service (DoS). In a DoS attack, an attacker seeks to overload a VM or a service within it, saturating its resources and making it unusable for legitimate users. This type of attack can cause disruptions to critical services and result in financial and reputational losses for the organization involved.

Another threat concerns data compromise within VMs through Data Breach attacks. Attackers can exploit vulnerabilities in applications or operating systems within VMs to access sensitive or confidential data. This data may include personal information, financial data, or company secrets. A data breach can have legal consequences and damage customer and user trust.

Man-in-the-Middle (MitM) attacks represent another threat to VMs. In this type of attack, an attacker inserts themselves into the communication between two legitimate parties, gaining access to the exchanged data. This type of attack can be particularly dangerous in virtualized environments where different VMs communicate with each other or with external systems. A MitM attack can compromise the confidentiality and integrity of communications.

Another attack concerns vulnerabilities within the VMs themselves, such as vulnerabilities in the operating systems or applications hosted within the VMs. Attackers can exploit these vulnerabilities to execute arbitrary malicious code, compromising the entire virtual environment. This type of attack underscores the importance of keeping both hypervisors and VMs constantly updated to protect against new vulnerabilities.

Finally, Evasion attacks aim to bypass the security measures implemented within VMs. Attackers may adopt techniques to hide their malicious activity or avoid detection by security systems. These attacks require constant vigilance and security measure updates to identify and mitigate new threats.

From the described attacks, it is evident that each of them aims to compromise at least one element of the CIA triad. The CIA triad, composed of Confidentiality, Integrity, and Availability, represents one of the fundamental concepts in the field of information security and information technology. These three pillars are the cornerstone principles for ensuring the protection and proper management of information and sensitive data in a wide range of contexts, from the corporate to the governmental and even personal spheres.

Figure 2. CIA triad

Confidentiality, the first component of the CIA triad, refers to the need to protect information from unauthorized access. This principle aims to ensure that only authorized individuals can access certain sensitive data, preventing unauthorized disclosures or theft of information. Confidentiality involves the implementation of authentication and authorization mechanisms, data encryption, and access controls to limit access only to those with legitimate rights.

Integrity is the second crucial element of the CIA triad and refers to ensuring that data and information maintain their accuracy and reliability over time. Unauthorized data manipulation can have serious consequences, leading to incorrect decisions, loss of trust, and potential damage. To ensure integrity, it is necessary to implement controls that prevent unauthorized alteration of data, such as digital signatures, checksums, and change logs.

The last component of the triad, Availability, emphasizes the importance of ensuring that information is accessible and usable when needed. This aspect is critical, especially in contexts where data unavailability can have significant impacts on business operations or service delivery. Availability requires the design of robust, redundant infrastructure and business continuity plans to minimize disruptions and ensure constant access to information.

It is important to note that these three principles are not always in perfect balance and can even conflict. For example, increasing controls to ensure confidentiality may impact information availability. Finding the right trade-off between confidentiality, integrity, and availability is a constant challenge in the field of information security and often requires a careful risk assessment and consideration of specific context needs.

To address these challenges and ensure the security of VMs and hypervisors, it is crucial to conduct detailed security assessments and implement preventive measures. In this context, the hardening process plays a crucial role. Hardening refers to a set of actions and measures taken to make a system more secure, enhancing its ability to resist unauthorized access attempts, exploits (i.e., codes or techniques designed to exploit specific vulnerabilities in software or operating systems), and other types of threats and attacks [1-2].

To perform the hardening process, structured approaches provided by recognized security assessment frameworks can be adopted. Some examples include:

- ISO (International Organization for Standardization) 27001: An international standard for information security management. It provides guidelines for establishing and maintaining an Information Security Management System (ISMS), which includes procedures for identifying and addressing vulnerabilities.
- OSSTMM (Open Source Security Testing Methodology Manual): Focuses on testing and evaluating security measures. This framework promotes a rigorous and methodical approach. OSSTMM consists of three classes and five channels: Telecommunications and Network Data Security Channels of the COMSEC class, Physical and Human Security Channels of the PHYSSEC class, and the Wireless Security Channel of the SPECSEC class [3].
- OWASP (Open Web Application Security Project): Provides guidelines and resources for developing secure software, with a particular focus on web applications. OWASP lists major vulnerabilities and offers advice on mitigating them.
- NIST (National Institute of Standards and Technology): Offers guidelines and best practices for various aspects of cybersecurity. NIST recommendations are widely adopted in corporate cybersecurity.

The application of these structured frameworks helps identify, assess, and mitigate vulnerabilities and security risks in computer systems. In particular, the use of OSSTMM can provide a systematic framework for conducting security assessments, ensuring an engineering and methodical approach.

The thesis activity involves the use of a Type 1 hypervisor. This represents a strategic choice to ensure efficiency, isolation, and security of virtualized resources within the university environment. In this case, Proxmox, which is a complete operating system based on Debian, a Linux distribution, proves to be a particularly interesting and appropriate solution for efficiently and securely managing VMs and possible containers.

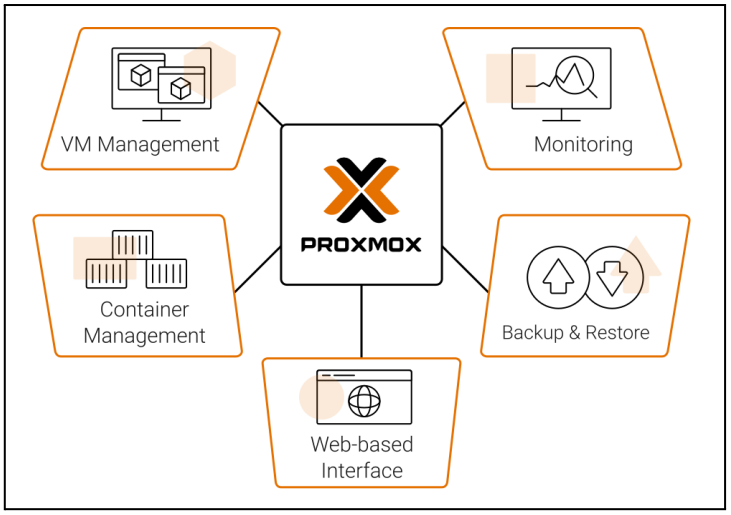

Figure 3. Proxmox features

Proxmox is based on advanced virtualization technologies, such as KVM (Kernel-based Virtual Machine) and LXC (Linux Container). The adoption of KVM as the underlying virtualization technology offers significant advantages in terms of performance and VM isolation. This is possible due to direct access to hardware provided by KVM-based VMs, allowing them to run instructions directly on the host CPU. The use of LXC containers, on the other hand, adds a lighter and more efficient level of virtualization, enabling the execution of isolated applications while sharing the same host system kernel.

The intuitive web interface provided by Proxmox greatly simplifies VM and container management. System administrators and users can leverage this user-friendly interface to create new VMs, clone them, allocate resources such as CPU, RAM, and storage, monitor real-time performance, and configure virtual networks. This ease of use is of great importance in ensuring proper configuration and effective resource allocation within the virtual environment.

The implementation of backup and restore features within Proxmox further contributes to data security and operational continuity. The ability to regularly back up VMs and containers, as well as the option to restore to a previous state in case of errors or incidents, is essential to mitigate the risk of data loss and unwanted interruptions.

However, a critical aspect is the lack of a security assessment and hardening process on the VMs managed within this platform. The absence of a thorough vulnerability analysis and an adequate hardening process can expose VMs to security risks that may be exploitable by external or internal attacks.

In this context, the application of security assessment frameworks like OSSTMM becomes crucial to address this issue. The use of OSSTMM will allow for a structured and methodical assessment of vulnerabilities in the VMs, identifying potential weaknesses and defining a plan of action for implementing hardening measures.
Furthermore, the adoption of Proxmox and the execution of a security assessment can be seen as an opportunity for the DIEM to strengthen the security of the entire virtual environment, ensuring the protection of sensitive data and operational continuity. The choice of an open-source hypervisor like Proxmox also demonstrates a sensitivity to widely monitored solutions that prioritize quality and security.

2.2. Identification of potential advancements beyond the state of the art

Technological advancement and the increasingly widespread adoption of virtualization infrastructure have introduced new challenges and opportunities in the field of cybersecurity. In the context of the thesis work, the analysis conducted on the state of the art concerning security assessment and hardening practices of hypervisors and virtual machines has revealed the need for a rigorous and systematic methodological approach to ensure a robust and resilient virtualization environment.
In response to this need, an innovative methodology has been developed for conducting security assessments and hardening of the provided systems. This methodology stands out from existing frameworks due to its emphasis on an iterative and replicable approach, enhancing its effectiveness and flexibility. It is important to note that the starting point is the OSSTMM, recognized for its conceptual soundness and comprehensive coverage.
A fundamental aspect of the innovation introduced by this methodology is its ability to address emerging challenges related to the virtualization environment. Compared to the state of the art, the application of this methodology results in a significant increase in the security of the entire virtualization ecosystem. This increase translates into greater ability to identify, prevent, and mitigate potential threats originating from both the hypervisor and the virtual machines hosted within it.
A key element of this methodology is its iterative nature, which allows for dynamic adaptation to changes in the cybersecurity threat context. Each assessment and hardening cycle incorporates the latest information on vulnerabilities, attacks, and countermeasures.

This proactive approach ensures that the virtualization environment remains resilient even in the face of continuously evolving threats.

Furthermore, a distinctive feature lies in its adaptability to various operational contexts. Whether it is a high-speed development environment or a high-security critical infrastructure, the methodology can be tailored and customized to meet the specific needs of each context. This flexibility is particularly valuable in today's rapidly evolving technological context.

In conclusion, the proposed process represents a significant advancement in addressing cybersecurity issues within any context. The synergy between the principles outlined in the OSSTMM guidelines, the systematic iteration adopted, and its practical application converges to create a tool aimed at optimizing the security of computer systems in an era characterized by a constant increase in digital threats.

Chapter 3. Original contribution to problem solution

3.1. Proposed methodology definition

The fundamental objective of this thesis activity is the design and implementation of a comprehensive and reliable process for conducting security assessment and hardening of computer systems. This process not only aims to ensure protection against threats and vulnerabilities in an increasingly complex and evolving environment but also seeks to demonstrate the practical validity of its approach through application in the DIEM-provided environment.

To achieve this ambitious goal, design choices were made that characterize the contribution provided. Specifically, the characteristics that should distinguish the proposed methodology were identified. Indeed, this methodology was developed with the goal of being:

- Structured: divided into phases, each of which plays a fundamental role in carrying out the activity. This structure allowed for clear and precise definition of the operations to be carried out to successfully apply the methodology.
- Generalizable: valid and applicable to any type of computer system.
- Iterative: capable of allowing a continuous cycle of assessment and improvement. After the implementation of specific security measures, activities must be repeated to evaluate the effectiveness of the solutions adopted and to identify new vulnerabilities or potential changes in the security context.
- Replicable and Consistent: when applied repeatedly without a change in context, it should produce the same results. This ensures the reliability of the methodology.

Furthermore, a careful study of the OSSTMM manual, developed by the ISECOM (Institute for Security and Open Methodologies), was conducted. This allowed for an understanding of the strengths of this standard and its integration into the methodology to be developed to achieve the predetermined characteristics.

The OSSTMM is based on a class-based approach, each of which provides a unique perspective on security. Specifically, there are three classes:

- COMSEC (Communications Security): This class focuses on communication security. It specifically analyzes and assesses the protection of sensitive information during transmission. This includes analyzing the integrity and confidentiality of data exchanged between network nodes, identifying possible interception threats or man-in-the-middle attacks, and adopting appropriate security measures to mitigate such risks.
- PHYSSEC (Physical Security): This class pertains to the physical security of computer systems. It considers aspects such as physical access to devices, protection against theft or physical intrusion, access control to sensitive areas, and the management of physical infrastructures. It ensures that systems are defended not only digitally but also against physical threats.
- SPECSEC (Specialized Security): This class covers specific and specialized security aspects that can vary depending on the context. For example, it could involve industrial security, specific vulnerabilities of IoT devices, or unique challenges related to embedded systems. The SPECSEC class adapts to particular scenarios, providing a framework to address unique and infrequent threats.

The approach supported by OSSTMM enjoys widespread acceptance and use within the cybersecurity community. Its inherently generic and flexible nature allows for adaptation not only to software and hardware but also to a wide range of contexts. This flexibility is a crucial feature, recognizing the increasing interconnection of modern systems and the need to address threats comprehensively.

It is important to note that OSSTMM is also open-source, promoting collaboration among experts and dynamic adaptation to emerging challenges in the security context. This open and collaborative approach helps keep OSSTMM relevant and effective in addressing increasingly sophisticated cybersecurity threats. It is worth noting that, at the time of writing this thesis, OSSTMM has reached its third version, demonstrating a continuous commitment to innovation and refinement of its methodologies.

After careful and in-depth study of the manual published by ISECOM, it was chosen to base the approach on this methodology, specifically focusing on the COMSEC class related to Data Networks. Based on this, a security assessment and subsequent hardening process was designed, articulated in various phases. These phases contribute to ensuring the security of the system. Specifically, the phases are:

1. Data collection & Mission definition
2. Environment setup
3. Vulnerability analysis
4. Exploitation
5. Outcomes evaluation
6. Hardening

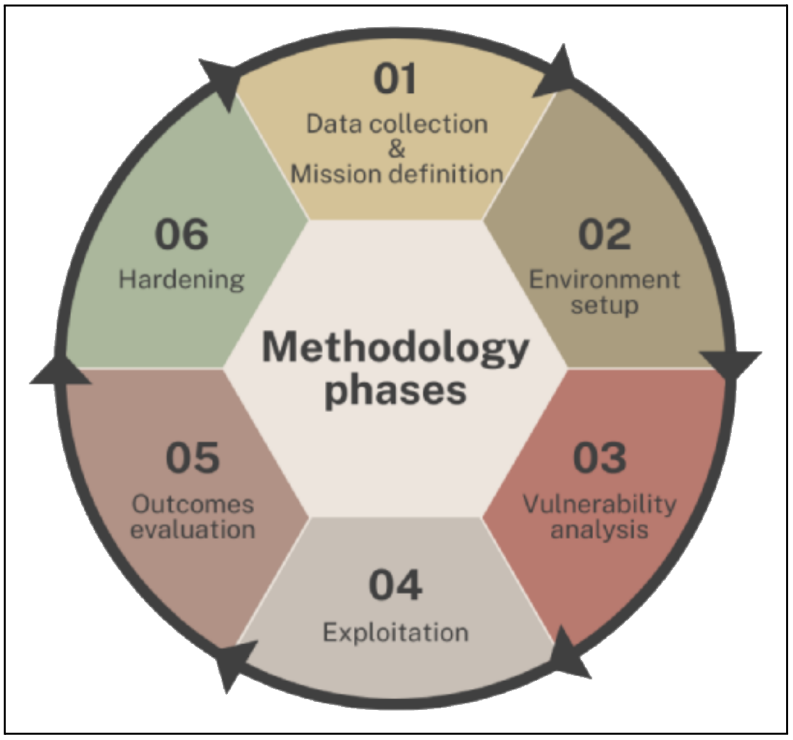

Figure 4. Methodology phases

3.1.1. Data collection & Mission definition

In the context of initiating a security analysis process, the data collection and mission definition phase plays a crucial role. Indeed, there are several interconnected components that require careful and strategic management to obtain reliable and relevant results. A key element in this process is proactive and ongoing communication with the stakeholders who have entrusted the assessment task. This dialogue is crucial for clearly defining objectives, understanding the specific needs of the involved parties, and gathering fundamental information about the operational context of the system under analysis. In this way, it is also possible to define the scope of the target network segment, as highlighted in the OSSTMM. This initial step provides a solid foundation upon which to build the entire assessment process and the actual level of knowledge with which to approach the activity.

Specifically, there are three types of approaches:

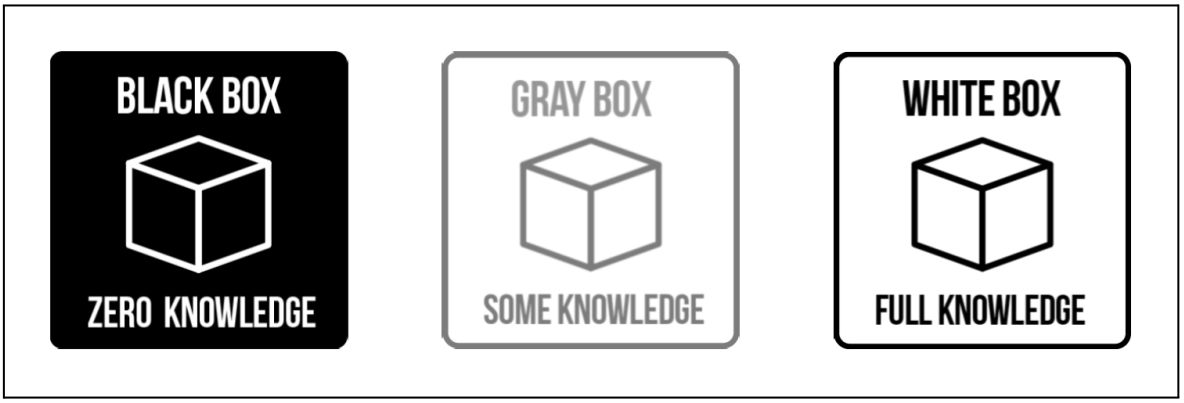

Figure 5. Knowledge approaches

- Black Box: In the black box approach, analysts operate without any internal knowledge of the system. This emulates the situation in which an external attacker seeks to exploit vulnerabilities without having access to the source code or internal details. Analysts rely solely on external interactions with the system and seek to identify vulnerabilities through penetration testing, fuzzing, and other techniques. This approach is useful for assessing how exposed a system is to external attacks and how resistant it is to common attacks. Furthermore, interaction with employees may be minimal or absent. Since analysts operate without internal knowledge of the system, the goal is to simulate an external attack. In this scenario, interaction with employees may be kept to a minimum or may not be necessary, as analysts focus primarily on external interactions with the system.
- Grey Box: The gray box approach falls between white box and black box. Analysts have partial knowledge of the system but do not have all internal details. This may include information such as high-level architecture diagrams, limited access to source code, or interface information. Analysts use this limited knowledge to simulate attacks that could be carried out by attackers with a certain level of knowledge. The grey box approach provides a good balance between the in-depth analysis of the white box and the real-world representation of the black box. Additionally, interaction with employees may be limited compared to the white box but is still present. Analysts may have access to high-level information or interact with selected personnel to obtain additional details about the system. This

collaboration may focus on specific areas that require further investigation without revealing the entire system's framework.

- White Box: In the white box approach, analysts have full access to the internal structure of the system under examination. This means they have access to the source code, documentation, and any other technical information needed. In other words, the system is examined in every part. This level of knowledge allows analysts to examine how the software was designed and implemented in detail. They can identify potential vulnerabilities in-depth and propose specific solutions. The white box approach is particularly useful for identifying structural or implementation security flaws. Furthermore, it is very likely that analysts can interact with employees who developed or are responsible for the system. Since analysts have complete access to the source code and internal structure, they may request detailed explanations about specific aspects. Collaboration with employees can be valuable in gaining a better understanding of the system's operation, identifying any vulnerabilities, and evaluating the security measures in place.

The choice between white box, gray box, and black box approaches depends on the goals of the security assessment and the amount of information analysts can have or simulate.

Time planning is an element that deserves careful consideration. Indeed, determining how much time to allocate to each phase of the assessment and assessing the possibility of acquiring new skills, if necessary, are critical aspects for the success of the project. Adequate timing allows for a detailed analysis and addressing challenges that may arise along the way.

This preliminary stage of the assessment process implies the precise and detailed formulation of specific objectives that serve as guidelines for the entire analysis procedure. In this perspective, the importance of accurately defining objectives cannot be underestimated.

The definition of objectives forms the foundation on which the entire security assessment process is built. Essentially, these objectives define the boundaries within which the assessment will take place and establish the underlying intentions that guide the work. These objectives not only provide a clear and articulated structure for the analysis path but also serve as a constant reference point throughout all phases of the process.

A carefully defined mission creates a coherent and rational framework for the entire evaluation. It unequivocally establishes what will be examined and what methodological approach will be adopted. This sets off a cascade effect throughout the process, as every

subsequent decision and step taken will be guided by these predefined objectives. Consequently, mission definition is not merely a formality but a critical process that will influence the validity and effectiveness of the entire assessment.

The clarity and specificity of mission objectives are crucial to avoid ambiguity and misinterpretations. Furthermore, they provide a solid basis for the strategic planning of assessment activities. The outlined objectives must be realistic, measurable, and aligned with actual security needs.

3.1.2. Environment setup

In the context of preparing the environment for security assessment, it is crucial to adopt a comprehensive and meticulous approach. This phase requires a series of actions to establish a solid foundation for the success of the entire process.

A crucial step in this phase is the identification of systems to undergo security assessment and subsequent hardening. These systems can be physical, virtual, or hybrid and should be selected based on their relevance to the project's context. For example, they may include the hypervisor used for virtualization, storage devices used to store sensitive data, virtual machines hosting applications, and virtual networks connecting these components.

Configuring the working environment to generate a replica of the real environment, including operating systems, applications, and configurations, is necessary to perform any type of security analysis. To carry out this operation, alternative physical or virtual machines can be created using backups of the real ones. This will enable conducting realistic and relevant security analyses.

Equally important is the configuration of networks, whether physical or virtual, of the machines used. They must be configured like real ones to ensure the reproducibility of communications within the test environment.

Finally, the selection and installation of security analysis and monitoring tools represent an essential step. These tools should be chosen based on the specific project requirements and configured to collect relevant data and detect potential security threats.

Within the context of OSSTMM, particular attention is given to the activities to be carried out in this phase. This methodology places special emphasis on the execution of specific and detailed activities aimed at ensuring a comprehensive and precise understanding of the assessment context.

A fundamental step in this phase is the identification of IP addresses of the machines involved in the assessment, if such information has not already been provided during the

previous phase. IP addresses are the key to establishing connectivity and accessibility to network resources and must, therefore, be accurately identified. This process may involve packet sniffing, which is the monitoring of network traffic to capture and analyze data packets in transit.

Once the IP addresses are acquired, a highly recommended activity is to verify their reachability using the ping operation. This allows you to check if the machines are active, online, and ready for examination. Confirming reachability through ping provides a solid basis for the actual execution of security analyses, as it ensures that the machines are reliably accessible.

In general, this phase, according to OSSTMM, requires careful planning and the implementation of detailed procedures to ensure that the working environment is ready for security assessment.

3.1.3. Vulnerability analysis

In the context of this phase, the crucial importance of fully leveraging all available resources to conduct a comprehensive and detailed investigation becomes evident. This process plays a fundamental role in cybersecurity as it aims to identify potential weaknesses in the systems or applications under examination. The methodology adopted reflects the principles outlined by OSSTMM, which emphasize the need for a meticulous and systematic approach to ensure reliable and meaningful results.

First and foremost, the initial step in this phase involves the enumeration of open ports. This operation, though it may seem simple, constitutes a foundational pillar for understanding the system's exposure to the external environment. It entails scanning and recording all active and accessible network ports, with particular attention to identifying ports used for specific services. This process allows for a precise delineation of how the system is connected to the network and which services are accessible to potential external threats.

A highly recommended practice is domain name mapping so that they are associated with specific machines. This step adds a significant level of understanding and traceability to the working environment. For example, instead of using only IP addresses to refer to machines, domain names can be used to identify resources more intuitively and legibly.

Once the enumeration of open ports is completed, the second step involves using tools and techniques to identify, classify, and quantify vulnerabilities present in the systems and

applications. These tools can range from automated vulnerability scanners to vulnerability detection services based on continuously updated databases.

The primary objective of vulnerability analysis is to accurately and comprehensively detect and document all identified vulnerabilities.

It is crucial to emphasize that the field of cybersecurity is constantly evolving, with new threats and vulnerabilities continuously emerging. Therefore, this phase, along with the entire proposed methodology, is an ongoing and dynamic process that requires constant vigilance. Periodic reviews and updates of security assessments are essential to ensure that the system remains protected over time. This constant pursuit of improvement is crucial to adapting to changing threats and maintaining the integrity and security of computer systems.

In conclusion, vulnerability analysis is a process of fundamental importance in the cybersecurity defense strategy of any organization, requiring advanced skills, attention to detail, and a mindset always ready to adapt.

3.1.4. Exploitation

After conducting a thorough analysis of the environment or machines undergoing security assessment, this phase aims to test the robustness of defenses. Essentially, it involves concretely demonstrating how vulnerable a system is to external or internal attacks.

To do so, it is necessary to exploit the vulnerabilities previously identified. These vulnerabilities can arise from misconfigurations, weak passwords, or the presence of bugs within software and operating systems. The primary objective of this phase is to demonstrate how a potential attacker could leverage these weaknesses to gain unauthorized access to systems or cause damage in some way.

One avenue to explore involves exploiting misconfigurations within the system. This can include the use of unchanged default configurations, neglecting critical security updates, or lacking hardening procedures. Attackers can exploit these situations to gain unauthorized access or compromise the system.

Furthermore, weak passwords often represent an easy entry point for attackers. Here, exploitation may involve attempts to guess or decrypt passwords used by users or administrators. Techniques like brute force or the use of password dictionaries are commonly employed. The desired outcome is unauthorized access to systems, highlighting the importance of using strong passwords and multi-factor authentication.

Finally, the use of exploits is a key aspect of this phase. Security operators can employ these tools to demonstrate the actual exposure of systems and the potential damage an attacker could inflict. Identifying and remedying these vulnerabilities is a fundamental element in safeguarding computer environments.

The primary objectives of an attacker include not only gaining access to the target machine but also the ability to exercise complete control over the system to perform malicious operations. To achieve these objectives, the security operator must undertake several significant actions by impersonating the attacker.

Firstly, they need to attempt to gain access to the system. This can be accomplished in various ways. Initially, one might seek to exploit specific vulnerabilities in the operating system or applications running on the machine. This could lead to unauthorized access. Alternatively, the use of credentials previously obtained through weak password searches or other information gathering methods may grant legitimate access.

Another option is opening a reverse shell. This technique enables the attacker to establish a remote connection to the system, often by exploiting specific exploits. Through this connection, the attacker can gain partial control over the machine, allowing for the execution of certain commands, access to files and directories readable by anyone, and the ability to interact with the system as if physically present, impersonating a user with limited privileges. This level of control is crucial for system analysis and uncovering further vulnerabilities or sensitive information.

After obtaining any form of access, if the attacker has not already secured an administrator account, the next step is privilege escalation, i.e., obtaining the highest possible permissions. This elevated level of access grants the attacker the power to modify any system configuration and make substantial changes. Privilege escalation is often a key objective as it allows the attacker to bypass security restrictions and gain full control over the system. From a security standpoint, it represents the most dangerous outcome as it can lead to significant damage or the installation of hidden backdoors, i.e., access points that can be exploited in the future.

3.1.5. Outcomes evaluation

Within this phase, the primary objective is to compile a detailed report that captures the essence of all the work done in the previous stages of the security analysis. This report plays a central role as it serves as the convergence point for all the data, information, and findings gathered throughout the assessment journey.

A fundamental aspect of this phase is the accurate documentation of all the vulnerabilities identified within the system under examination. For each of these vulnerabilities, the following information is reported:

- Vulnerability source: It is essential to clearly specify which element of the system is vulnerable.
- Potential impact: It is crucial to provide an assessment of the potential impact this vulnerability could have on the overall system security.
- Exploitation likelihood: It is important to assess the likelihood with which this vulnerability could be exploited.
- Expected consequence: It is fundamental to specify clearly what the objective would be if this vulnerability were to be exploited.

A crucial step in this phase is conducting a risk analysis for each identified vulnerability. When referring to risk, it encompasses the combination of the probability of a vulnerability being exploited and its potential impact.

In this case, the objective is to assign a numerical value to each vulnerability to quantify the associated risk. This process involves the use of metrics and assessment criteria that allow for comparing vulnerabilities and establishing priorities based on the degree of danger they pose to the system.

Often, accurately determining probabilities and realistic cost consequences related to risks can be challenging due to various factors, including the complexity of situations and the inherent uncertainty surrounding future events.

To address this challenge, qualitative methods are often used to assess risks. These methods rely on a more subjective rather than quantitative evaluation and use a rating scale to prioritize risks and identify those requiring immediate attention.

A common approach is to use tables representing the probability of a threat occurring and its potential impact if it does. These tables contain a range of levels, such as "very low" to "very high" for probability and "negligible" to "catastrophic" for impact.

It is then possible to qualitatively evaluate the probability and impact of each risk, using these levels as a guide. Moreover, it allows focusing on managing the most evident and significant risks even in the absence of precise quantitative data.

In addition to these essential operations, the OSSTMM suggests calculating an index called RAV (Risk Assessment Value). This can be obtained through the use of a comprehensive and well-structured document and represents an overall assessment of the system's attack surface. Specifically, an RAV value of 100 represents a perfect balance, while a lower value implies a larger attack surface and a higher value indicates an unnecessary number of controls. The RAV provides a strategic perspective on security, helping to understand the scope of risks and make informed decisions regarding the implementation of security measures.

Specifically, within the RAV, information is required, grouped into eighteen categories divided into three areas.

Within the Operational Security area, the following categories are present:

- Visibility: the number of targets within the analysis context.
- Access: the number of points through which interaction can be established. In the context of COMSEC analysis, it represents the number of open ports with an active service.
- Trust: the number of trust relationships between targets within the analysis context. In the context of COMSEC analysis, it represents the number of forward services and port forwards.

Within the Controls area, the following categories are present:

- Authentication: the number of authentication instances required to gain access.
- Indemnification: the number of methods used to establish responsibility and ensure compensation for assets within the analysis context.
- Resilience: the number of instances designed to be fail-safe, meaning they do not grant access in case of errors.
- Subjugation: the number of instances that do not strictly require controls to follow user discretion.
- Continuity: the number of instances that ensure no interruption can occur in interactions, even in cases of complete failure. In the context of COMSEC analysis, it represents the number of services that offer redundancy so that no interaction is lost.

- Non-repudiation: the number of instances that prevent users from denying certain interactions. In the context of COMSEC analysis, it represents the use of multiple log files that record interactions.
- Confidentiality: the number of instances that provide means to keep interaction content confidential.
- Privacy: the number of instances that provide means to keep access methods, viewing, or data exchange confidential.
- Integrity: the number of instances that provide means to control changes. In the context of COMSEC analysis, it can be the use of file encryption or hashing that can provide integrity control over file changes in transit.
- Alarm: the number of instances that provide means to record interaction occurrences and potentially send appropriate notifications. In the context of COMSEC analysis, it represents the use of interaction log files.

Within the Limitations area, the following categories are present:

- Vulnerability: the number of errors that challenge protections in a way that a person or process can access, deny access to others, or hide or obscure assets. In the context of COMSEC analysis, it represents a defect that allows an attacker to gain privilege escalation, execute arbitrary code, or cause a denial of service.
- Weakness: the number of errors in controls for authentication, indemnification, resilience, subjugation, and continuity. In the context of COMSEC analysis, it represents a flaw like the ability to make unlimited attempts for access.
- Concern: the number of errors in controls for non-repudiation, confidentiality, privacy, integrity, and alarm. In the context of COMSEC analysis, it represents an issue like the use of locally generated web server certificates for HTTPS or log files that record only interaction participants but not the correct date and time it occurred.
- Exposure: the number of errors that provide direct or indirect visibility of targets or assets within the analysis context.
- Anomaly: the number of unidentifiable or unknown elements that cannot be explained in normal operations within the analysis context.

As you can see, and as explained previously, special attention has been given to the Data Networks channel of the OSSTMM COMSEC class.

This phase represents the culmination of the security analysis process, transforming raw data into crucial information for system protection. This comprehensive and methodical

approach allows us to fully understand the nature of threats, assess their impact, and define a mitigation strategy based on solid and objective data.

3.1.6. Hardening

This represents the final phase of the proposed methodology, which plays a crucial role in ensuring the security of the environment and the involved machines. In this phase, the main objective is the implementation of the most advanced and context-appropriate security measures to make the environment as resilient as possible against potential threats and cyberattacks. Identifying and applying these security measures are a crucial step in protecting both the infrastructure and sensitive data.

Among the security measures that may be considered in this phase, the following can be mentioned:

- Software and system updates: Ensuring that all components are up to date with the latest security patches can automatically eliminate many vulnerabilities. This is possible thanks to the work of manufacturers who should have addressed such issues.
- Firewall usage: These tools act as filters, allowing only the passage of legitimate packets while blocking anything that appears suspicious or irrelevant according to the set rules.
- File permission control: It is fundamental to verify and limit file permissions, especially those related to configuration files. Ensuring that only authorized users and processes have access to configuration files reduces the risk of unauthorized manipulations that could compromise system security.
- System access policy management: Enforcing strict and controlled access policies is essential. This means assigning specific permissions and roles only to certain users, avoiding unnecessary or privileged access. Managing access policies helps ensure that only authorized individuals can access system resources.

Once the hardening phase is complete, it is common practice to periodically repeat the entire proposed methodology. This continuous security assessment cycle serves to conduct a new security assessment and compare the final documentations produced with the previous ones. The goal is to detect whether there has been a potential reduction in the number of vulnerabilities in the system, which could result in a higher RAV, indicating a smaller attack surface and overall improvement in the security of the environment and the involved machines.

In essence, this continuous assessment cycle confirms the effectiveness of the adopted security measures and ensures the continuity of protection in an ever-evolving environment. Cybersecurity is a dynamic process that requires constant vigilance and adaptation to evolving threats, and this final phase plays a significant role in that effort. Protecting the organization's data and resources is a top priority, and the described methodology plays an essential role in achieving this goal.

3.2. Virtual machines and used tools

To implement the proposed methodology, it is necessary to use a wide range of technologies and tools to automate vulnerability and threat research. This aspect is crucial to ensure a comprehensive and effective analysis.

To facilitate the planned analyses, a dedicated virtual machine running Kali Linux has been adopted. Kali Linux is a Debian-based distribution specifically designed for penetration testing and security assessment activities. The choice of Kali Linux as the operating environment was made with careful consideration of several factors contributing to its suitability for this project.

Firstly, Kali Linux is known for being a cutting-edge distribution in the field of cybersecurity, with a vast ecosystem of preinstalled and readily available tools. This means that the tools required to assess the security of other systems are already present and configured in the virtual machine, significantly reducing the time and effort required to set them up individually and manually.

Additionally, this operating system is known for being constantly updated to address new threats and vulnerabilities, which is essential when working in the field of cybersecurity, where threats are constantly evolving.

For conducting the analyses on the systems provided for the thesis activity, the following multiple tools were used:

- Nmap: An acronym for "Network Mapper", it is a powerful and versatile cybersecurity tool used for analyzing and scanning computer networks. This tool is widely employed in the field of network security and IT infrastructure management. Its primary function is to perform a detailed mapping of a network, identifying connected devices, open ports, and services running on those devices. Nmap can detect network topology, the presence of vulnerabilities, and network device configurations. Its importance in cybersecurity lies in its ability to allow

system administrators and security experts to identify potential weaknesses in the network, such as unauthorized open ports or unnecessary services, which could be exploited by potential attackers. Nmap is an extremely flexible tool that can adapt to specific user needs through the use of customizable options and scripts. Additionally, it provides detailed network configuration information, facilitating security planning and cybersecurity strengthening.

- Dirb: An acronym for "Directory Buster", it is an important tool used in cybersecurity and network management. It is a specialized tool designed to identify and enumerate directories and files on a web server. Its main function is to systematically scan a website to discover hidden or unauthorized directories that could potentially be points of vulnerability or provide sensitive information. Dirb is widely used by cybersecurity experts and system administrators to assess the robustness of a website and ensure that all directories are correctly configured in terms of permissions and accessibility. This tool can be used to conduct penetration tests and security assessments, allowing the identification of any weaknesses in web server configuration or directory management. Its importance in the context of cybersecurity lies in the need to protect websites from potential cyberattacks, including unauthorized access attempts or attempts to access sensitive information. Dirb provides security operators with an effective means to identify and address potential weaknesses, thereby improving the website's resistance to cyber threats.
- SecLists: An acronym for "Security Lists", it is a valuable resource in the field of cybersecurity, known for being a vast collection of wordlists and datasets containing data useful for security experts, researchers, and developers. This collection includes various keyword lists, password lists, dictionaries, exploit patterns, reference data, and other relevant information for conducting penetration tests, security assessments, and computer-related research activities. The primary goal of SecLists is to facilitate the identification and mitigation of vulnerabilities in systems and applications. Word lists and information contained in SecLists are often used in conjunction with password cracking tools, vulnerability scanners, and other security software. The importance of SecLists in cybersecurity lies in its ability to provide reliable and up-to-date resources for experts and professionals working to protect digital assets and identify potential threats. Access to a wide range of word lists and reference data is essential for conducting realistic

penetration tests, improving the security of applications and systems, and ensuring an effective response to cybersecurity challenges.

- FFuF: An acronym for "Fuzz Faster U Fool", it is an essential tool in cybersecurity and penetration testing and is known for its ability to discover hidden directories and files on web servers. This application plays a crucial role in the assessment of the security of web applications and the discovery of potential vulnerabilities. FFuF's primary function is to automate the scanning of a directory on a web server, attempting to locate directories and files that may not be immediately visible or accessible to the average user. Using a list of possible directory and file names, FFuF makes HTTP requests to the server, analyzing responses to determine if the sought-after resources exist or not. This process can reveal hidden directories, sensitive configuration files, or even security vulnerabilities. The importance of FFuF in cybersecurity lies in its ability to reveal hidden information that could be exploited by attackers to compromise an application or system. Finding unauthorized directories or files can be key to identifying potential weaknesses or vulnerabilities that require immediate attention from system administrators or web developers.
- WPScan: An acronym for "WordPress Scan", it is a specialized cybersecurity tool used to assess the security of WordPress installations. WordPress is one of the most widely used Content Management Systems (CMS) globally, making it an attractive target for cyberattacks. WPScan was created to assist cybersecurity experts and system administrators in evaluating and strengthening the security of their WordPress platforms. WPScan's main function is to conduct a comprehensive scan of a WordPress website, identifying known vulnerabilities, outdated plugins, or potential configuration issues. It uses a database of known and updated vulnerabilities to compare the version of WordPress and installed plugins with publicly known vulnerabilities. This way, WPScan provides security operators with crucial information to address potential threats. The importance of WPScan in the context of cybersecurity lies in the need to protect WordPress platforms, often used to manage websites of various kinds, from possible cyberattacks and breaches. WPScan allows system administrators and security experts to identify weak areas and vulnerabilities, ensuring a timely response to mitigate risks and strengthen the website's security.

- Nikto: It is a widely used tool in cybersecurity and penetration testing, designed to assess the security of web servers. Its main function is to perform an automated scan of a web server to identify potential vulnerabilities and configuration issues. This tool is particularly useful for system administrators and cybersecurity experts as it helps detect and address potential security threats to websites and servers. Nikto operates by analyzing various aspects of a web server, including the server's configuration, hidden or unauthorized files and directories, running software versions, and known vulnerabilities. Using a database of vulnerability signatures and recognitions, Nikto can compare scan results with a list of known vulnerabilities, reporting any detected matches. This allows security operators to take timely action to correct and mitigate potential threats. The importance of Nikto in cybersecurity lies in its ability to automate and simplify the web server security assessment process. System administrators can use Nikto to identify potential weaknesses and vulnerabilities, improving the overall security of the server. Additionally, Nikto is an open-source tool, which means it is accessible and customizable to meet specific organizational needs.
- LinPEAS: An acronym for "Linux Privilege Escalation Awesome Scripts", it is an essential cybersecurity tool for analyzing and identifying vulnerabilities in Linux systems. Primarily designed for penetration testing and security checking, LinPEAS is an extension of the well-known PEAS tool and specifically focuses on enumeration and finding privilege escalation opportunities in Linux-based systems. LinPEAS's main function is to perform a comprehensive scan of the system, gathering a vast amount of information about configuration, running processes, services, system files, and permissions. This information is then used to identify potential weaknesses and vulnerabilities, often associated with possible privilege escalation. LinPEAS automates this detection process, greatly simplifying the work of security experts. The importance of LinPEAS in cybersecurity lies in the assistance it provides to system administrators and security experts in identifying and addressing vulnerabilities before they can be exploited by potential attackers. Privilege escalation is a common goal of cyberattacks, and LinPEAS plays a crucial role in detecting opportunities for such attacks, contributing to the strengthening of Linux system security.
- Hydra: It is an essential cybersecurity tool in the field of penetration testing and network security analysis. It is a powerful password cracker and brute force tool

designed to test the strength of passwords and identify potential vulnerabilities in systems that require authentication. Hydra's main function is to automate the process of login attempts. This method involves using an extensive list of possible passwords or combinations to attempt to guess access to an account or service. Hydra can be configured to perform brute force attacks against various authentication protocols, including SSH, HTTP, and many others. The importance of Hydra in cybersecurity lies in its utility for security experts and system administrators to identify potential weaknesses in passwords used in systems and services. Revealing weak or vulnerable passwords can help improve the overall security of a system by encouraging the adoption of stronger passwords and better management of access credentials.

Chapter 4. Experimental validation and practical applications

4.1. Description of data and case study used for experimentation

To conduct the thesis activity, as previously discussed in the context of the state of the art, the DIEM has provided a dedicated server. This server is equipped with the type 1 hypervisor Proxmox in its version 7.4-3. The choice of Proxmox was motivated by its proven reliability and the numerous advanced features it offers.

In terms of hardware, the server represents a powerful platform for running virtual machines and conducting the experiments required for the proposed methodology. The hardware specifications include two hexa-core sockets, providing a total of 12 CPUs, which offer considerable computational power to support the needs of virtual machines. Additionally, the server is equipped with a generous amount of RAM, totaling 40GB, allowing us to run multiple instances of virtual machines simultaneously without compromising overall performance. In terms of storage space, the server has approximately 950GB of storage, which has been allocated thoughtfully. Specifically, 45GB of space has been reserved for storing ISO images, while the remaining space has been allocated for running the virtual machines, ensuring that they have sufficient resources to operate without impediments. This hardware ensures a stable and scalable development environment for carrying out the activities.

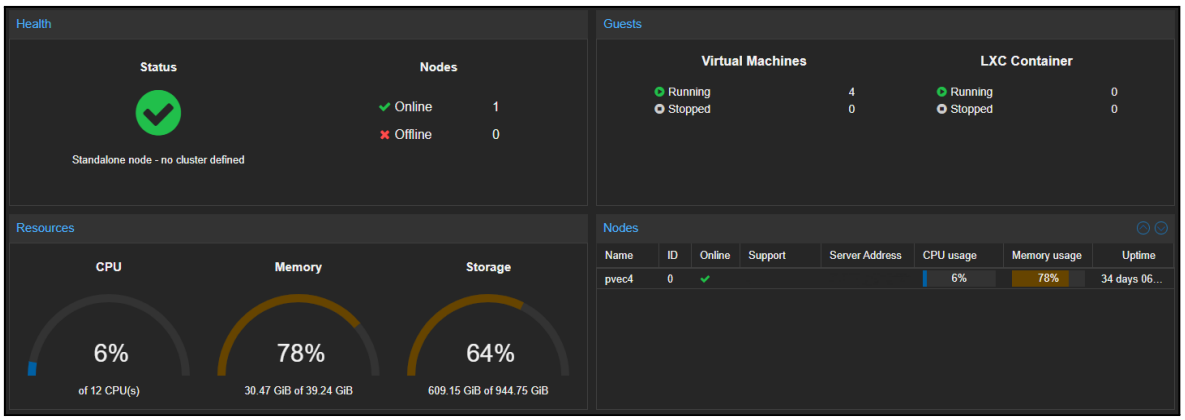

Figure 6. Server specifications

Within the server, virtual machines play a fundamental role in conducting security assessment operations and subsequent implementation of hardening measures. Each virtual machine represents a faithful replica of a physical machine actively used within the university context. They have been created with the aim of accurately reflecting the real operational environment, enabling realistic testing and assessments.

In particular, the three virtual machines within Proxmox are configured based on the specifications of their respective real machines:

- The VM dedicated to the Robotcup platform, which has two dual-core sockets for a total of 4 CPUs, 8GB of RAM, and approximately 120GB of storage space.

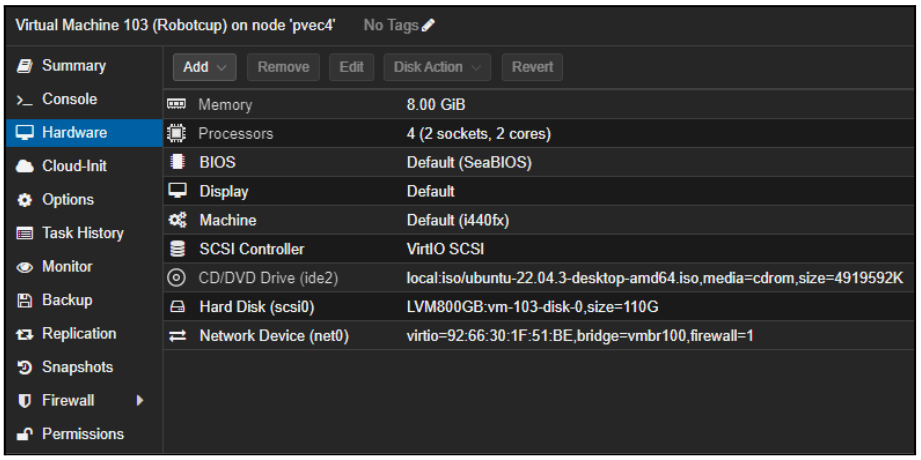

Figure 7. Robotcup specifications

- The VM dedicated to a platform where teachers can input their personal information, contacts, and websites, which has one dual-core socket for a total of 2 CPUs, 4GB of RAM, and approximately 23GB of storage space.

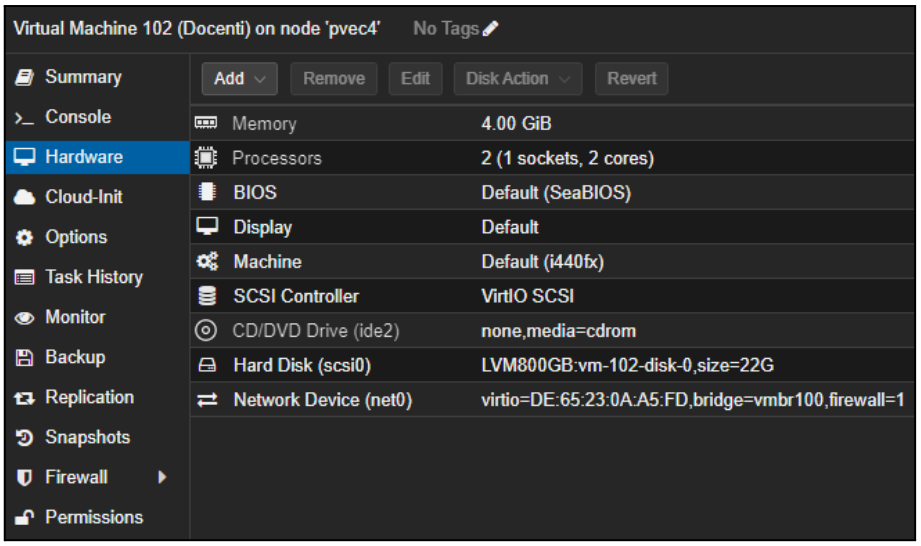

Figure 8. Docenti specifications

- The VM dedicated to the Gitlab repository of the MIVIA Lab at the University of Salerno, which has one dual-core socket for a total of 2 CPUs, 16GB of RAM, and approximately 400GB of storage space.

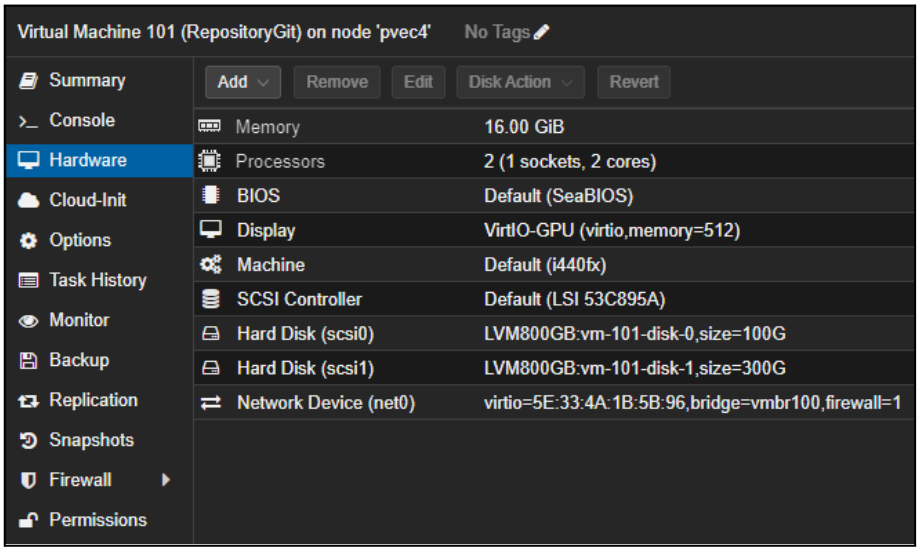

Figure 9. RepositoryGit specifications

To ensure secure access to the hypervisor and all VMs within the research environment, the DIEM has provided a Virtual Private Network (VPN). This VPN, accessible via the FortiClient VPN software, provides a secure communication channel with the university's network. Authorized users can use their institutional email credentials to establish a secure connection and remotely access the server.

The implementation of a VPN ensures the confidentiality and integrity of communications between authorized users and virtual resources, contributing to maintaining a secure and protected working environment throughout all phases of the thesis activity.

4.2. Methodology application

In the following paragraph, the methodology designed for conducting the security assessment and subsequent hardening of the provided environment and virtual machines, as described earlier, is applied. It is worth noting that the first two phases, namely "Data Collection & Mission definition" and "Environment setup", are common to all systems as they constitute preliminary phases in the context of the thesis activity.

4.2.1. Data collection & Mission definition

As part of the designed methodological approach, the initial phase of work involved interacting with the owner of the machines intended to undergo the security assessment and subsequent hardening process. This dialogue played a crucial role in establishing the necessary foundations for the thesis activity, including the definition of objectives to pursue.

Regarding the chosen approach, a black-box approach was decided upon. This decision was driven by the limited information available, which only included the subnet in which the virtual machines are located and their names. These details, while useful, provide only a very limited view of the activities performed by these machines in reality. Consequently, the black-box approach was considered the most appropriate for evaluating security.

It should be noted that, even though administrative user credentials will be used for accessing the Proxmox virtualization environment, the actions to be taken to implement the proposed methodology will always follow a black-box approach.

As for the definition of objectives, three distinct ones were identified, each focused on a fundamental aspect of cybersecurity. These objectives provided a solid and organized structure for the thesis activity, ensuring that every critical aspect of security was adequately addressed and analyzed throughout the project. This approach allowed for a comprehensive and in-depth approach to security assessment and the implementation of necessary hardening measures.

The first objective emphasizes the crucial importance of identifying relevant vulnerabilities within the systems, networks, or applications under evaluation. This objective goes beyond

the mere identification of obvious and trivial weaknesses. Instead, it focuses on identifying potential access points that could be exploited by malicious actors or digital threats.

This phase of vulnerability research and identification requires a careful and meticulous approach. It's not just about detecting obvious weak points but systematically delving deeper to uncover possible hidden or less apparent openings that could be used as unauthorized entry points. This methodology goes beyond the surface and delves into the complexity of systems, networks, and applications to uncover potential attack vectors that may escape superficial observation.

Addressing this challenge requires in-depth knowledge of communication protocols, involved technologies, and potential attack techniques that attackers might employ. One needs to be well-informed about tactics that could be used to exploit vulnerabilities, such as software vulnerabilities, misconfigurations, design flaws, and coding errors.

Thorough vulnerability analysis is essential to ensure a comprehensive and accurate security assessment. The goal is to highlight potential risks that may not be immediately evident but could be exploited by targeted attacks or more advanced threats. This identification phase requires dedication, technical expertise, and a problem-solving mindset to discover all possible weaknesses.

The second objective is the critical evaluation of the currently implemented security policies. This entails a thorough examination of the protective measures that have been adopted with the aim of determining their actual effectiveness and adequacy in addressing threats and potential attacks. This evaluation aims to identify possible gaps or deficiencies in the current security policies, as well as aspects that require improvement. Furthermore, it represents a crucial opportunity to optimize security strategies and adapt them to the evolving threats that characterize the dynamic context of cybersecurity.

Evaluating existing security policies requires a methodical and systematic approach. This process involves a detailed analysis of each component, such as access policies, authentication controls, password management procedures, and data encryption policies.

The main purpose of the evaluation is to determine the effectiveness of the security measures in place. This includes examining their practical implementation and adherence by users and administrators. Additionally, it seeks to identify potential risk areas that may not have been covered by current policies or that may be exposed to emerging threats.

The third and final objective focuses on identifying and implementing the best hardening practices. This phase, with the assistance of the previous one, requires a careful and meticulous analysis of system configurations, access policies, and management procedures.

The goal is to ensure that these aspects align with the most up-to-date and rigorous security standards. Best hardening practices not only significantly contribute to mitigating the risk of attacks but also represent an ongoing commitment to maintaining the IT environment as resilient as possible against emerging threats.

Identifying such practices requires a careful assessment of the specific requirements of the environment in question. Configurations and access policies must be tailored to the organization's needs, taking into account the applications used, sensitive data, and potential threats.

The objectives of vulnerability identification, evaluation of adopted security policies, and identification of best hardening practices work in synergy to create a comprehensive picture of the current security situation. This goal-setting process not only outlines the path to follow during the assessment but goes beyond, influencing the overall security strategy of the organization. The result is a strategic and comprehensive approach to protection that aims to ensure effective and proactive defense against the evolving and increasingly sophisticated cybersecurity threats.

This integrated approach results in a genuine understanding of threats and vulnerabilities, allowing the organization to act proactively rather than reactively. Security strategies are no longer viewed as mere requirements but as powerful tools to preserve the integrity, confidentiality, and availability of digital resources.

4.2.2. Environment setup

To successfully complete the thesis activity, it was essential to plan and organize the working environment adequately. In particular, as previously described, three virtual machines were provided within the Proxmox virtualization environment, located on a server owned by the DIEM at the University of Salerno.

To access this environment, a meeting was organized with the technician and administrator responsible for the infrastructure. During this meeting, essential information was provided to establish a connection to the dedicated server. Specifically, the IP address and port for web interfacing were communicated. Furthermore, access credentials for Proxmox with administrator privileges were provided to effectively manage the virtual machines and the environment.

Another crucial aspect learned during the previous phase is the subnet in which the three virtual machines were placed. It's important to emphasize that this subnet is distinct from

the one in which the Proxmox server itself is located. This distinction is essential because this is a virtualized environment.

Before proceeding with the analysis, it was deemed of paramount importance to conduct a detailed study of the virtualization environment. This phase was crucial to gain in-depth understanding of how virtual machines, storage space, and all components within the environment function and are accessible through the Proxmox user interface.

A critical aspect of this study was to determine the remaining available space on the server. This assessment was necessary to ascertain the presence of sufficient space to host a new Kali Linux virtual machine, a crucial step for conducting the subsequent phases of the thesis activity.

Once space availability was confirmed, the ISO image of the latest version of Kali Linux was downloaded from the official website and inserted into the virtual machine for booting. This step was essential to have a complete operational environment for conducting security assessments since Kali Linux comes with a wide range of preinstalled security and ethical hacking tools.

The configuration of this VM was designed to ensure high performance during the analysis operations, featuring two quad-core sockets for a total of 8 CPUs, 4GB of RAM, and 64GB of storage.

Creating the VM alone was not sufficient for its use in the following phases. It was necessary to place it on the same subnet used for the other provided virtual machines. To do this, the same Virtual Machine Bridge (vmbr) used by the other machines was utilized. These are interfaces for creating a virtualized network infrastructure, essential for ensuring communication and isolation of network traffic between the VMs hosted on Proxmox. This approach provided a flexible means to manage networks within the virtualization environment.

In particular, a vmbr representing a virtual phantom network was used. Subsequently, within Kali Linux, the interface associated with it was configured to assign it an appropriate IP address. This choice was made randomly by selecting an address from those available (a total of 255) in the hope that it would not conflict with those of the other three virtual machines to be analyzed.

Considering that the Proxmox configuration described above limited access only to those who connected to the previously provided IP address and port, with geographical restrictions to certain areas of the DIEM, a significant decision was made. In collaboration with the professors and the network infrastructure administrator, a fundamental change was

made to the system's access mode, making it available from anywhere in the world through the implementation of a department-owned VPN, as described earlier.

This change also entailed assigning a different IP address, while retaining the same port, to access the Proxmox interface. This decision opened a global access channel to Proxmox through the VPN, enabling secure and controlled remote connections. It was a strategic move to facilitate thesis work, providing reliable remote access to the virtualization environment regardless of geographic location.

To allow internet browsing and the ability to download additional tools or updates for those already present within the Kali Linux virtual machine, further network environment configuration was performed. Specifically, a new network interface, a new vmbr, was created and configured to obtain an IP address dynamically through Dynamic Host Configuration Protocol (DHCP) via the VPN.

DHCP is a widely used network protocol that allows devices to automatically obtain the necessary network configuration information for communication. It's designed to simplify and automate the process of assigning IP addresses and other network information, such as subnet masks, default gateways, DNS servers, etc. This protocol is particularly useful within Local Area Networks (LANs) and environments with many devices that require temporary or dynamic IP addresses, as it greatly simplifies IP address management in a network by allocating addresses to devices only when needed, thus avoiding waste. Furthermore, it facilitates device configuration and maintenance in the network, as much of the process is automated.

This additional interface was essential to enable the Kali Linux virtual machine to communicate with the outside world through the VPN. The use of DHCP simplified the configuration by allowing it to automatically obtain a valid IP address within the department's network. This ensured a stable and reliable internet connection, allowing the virtual machine to access online resources and download the necessary analysis tools and software updates without issues.

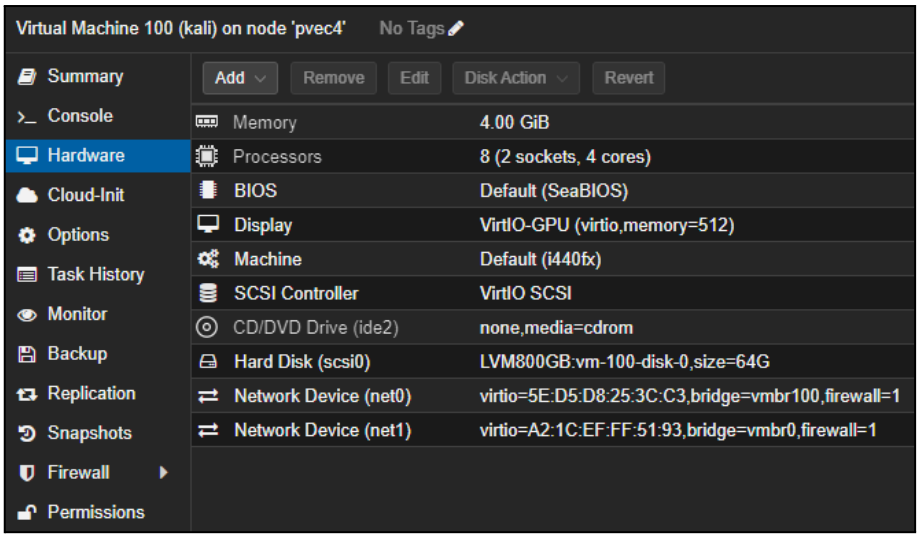

Figure 10. Kali Linux specifications

The preparation of the Kali Linux VM also included the installation of SecLists, as described earlier. This ensured the ability to use the wordlists present in that archive in combination with other tools.

To complete the configuration of the Kali Linux virtual machine, the Nmap tool, described earlier, was used to conduct a comprehensive scan of the provided subnet to identify the machines within it. From this analysis, in addition to the Kali Linux VM itself, three other virtual machines were identified, each with its unique IP addresses. This result was particularly relevant as it confirmed that the randomly chosen address for the VM in use did not conflict with those already present.

To verify the correctness of the obtained IP addresses, a reachability test was conducted using the ping command, which was successful. This step was crucial to confirm the connectivity and reachability of the identified machines.

Regarding the Proxmox virtualization environment, it's important to note that it was also considered a reachable machine from the Kali Linux VM. However, given the location of this environment within the university network and the prior knowledge of IP address and associated domain name information, a different strategy was chosen than using the Nmap tool for scanning. Specifically, a mapping between the IP address and domain name was added to the /etc/hosts file. This operation was performed to enable future searches based on the domain name rather than the numeric IP address. This approach made the identification of the machine more readable and intuitive and simplified the reference to network resources within the system.

Subsequently, the association between a carefully chosen keyword and the IP address was added to the /etc/environment file. This contributed to increased convenience and efficiency when executing commands in the subsequent analysis phases. In fact, thanks to this association, it was sufficient to use the keyword preceded by the "$" character, denoting the call to an environment variable, to refer to the domain name. This approach significantly simplified the syntax of commands during analysis activities.
As for the other machines, currently, there is no knowledge of their domain names. For this reason, if additional information is obtained from subsequent phases, their respective new entries will be added to the previously mentioned files.

4.2.3. Virtual environment: Proxmox

4.2.3.1. Vulnerability analysis

In the context of vulnerability analysis, various scans were conducted through the Kali Linux virtual machine to assess the security of the Proxmox environment.
The first crucial activity involved identifying open ports. These ports can potentially serve as entry points for external attacks, making it essential to conduct a thorough scan to identify them. To perform this analysis, the Nmap tool was used once again, but this time the scan was focused on the individual host rather than the entire subnet. This choice allowed for a detailed view of the situation.
Port scanning was divided into two main types: a less noisy scan and a more noisy scan. In this context, "noise" refers to the intensity level of the scan, which is determined by the quantity of packets sent and received by the tool.
In the less noisy scan, an effort was made to maintain a discreet profile by sending a limited number of packets to avoid arousing suspicion or negatively impacting the performance of the host under analysis. This mode is particularly suitable when conducting a stealth scan or when minimizing the impact on the network and the target system.
In detail, the less noisy scan was designed to minimize visibility on the network. This was achieved by sending SYN TCP packets without completing the connection, thus avoiding establishing real connections with the target system's ports. Furthermore, this scanning mode omitted the sending of ping packets, commonly used to check host availability on the network. However, it's important to note that while this strategy, useful for discretion, could compromise the accuracy of the results obtained. The lack of an actual connection may prevent the collection of detailed information about the services running on each port.

On the other hand, the more noisy scan was designed to maximize the amount of collected data. This approach involved using various options and features of Nmap to provide a comprehensive and detailed scan. In particular, this mode included:

- Scanning of all possible ports: This involved examining all available ports on the target, without omitting any possibilities. This may reveal open ports that could otherwise escape a less detailed scan.
- Operating system detection: Nmap attempted to identify the operating system of the target machine based on specific characteristics of responses to scan packets.
- Service version identification: Nmap tried to determine the versions of services running on open ports, allowing the detection of vulnerabilities associated with outdated or unpatched versions.
- Execution of default scripts: Default Nmap scripts were executed to gather additional information about configurations and potential vulnerabilities.
- Packet path tracing: This mode allowed tracking the packet path from the source machine to the destination host, providing a better understanding of the network topology.
- Latency measurement: The latency between the source machine and each hop along the path was evaluated, providing information on packet round-trip times.

This more detailed approach allowed for a complete picture of the Proxmox environment. However, it should be noted that such an intensive scan can be more invasive and may require more time to complete, in addition to potentially generating increased network traffic and being more detectable by security systems. Therefore, the choice between a less noisy and a more noisy scan should be based on specific objectives and security considerations.

The most significant result obtained from the scans was the identification of an open port associated with the SSH service, revealing the use of OpenSSH version 8.4p1. This information is particularly relevant because this version of OpenSSH is known to be subject to several documented security vulnerabilities.

The vulnerabilities identified are:

- CVE-2021-41617: sshd in OpenSSH 6.2 through 8.x before 8.8, when certain non-default configurations are used, allows privilege escalation because supplemental groups are not initialized as expected. Helper programs for AuthorizedKeysCommand and AuthorizedPrincipalsCommand may run with privileges associated with group memberships of the sshd process if the configuration specifies running the command as a different user.
- CVE-2021-36368: An issue was discovered in OpenSSH before 8.9. If a client is using public-key authentication with agent forwarding but without -oLogLevel=verbose, and an attacker has silently modified the server to support the None authentication option, then the user cannot determine whether FIDO authentication is going to confirm that the user wishes to connect to that server or that the user wishes to allow that server to connect to a different server on the user's behalf.
- CVE-2021-28041: ssh-agent in OpenSSH before 8.5 has a double free that may be relevant in a few less-common scenarios, such as unconstrained agent-socket access on a legacy operating system or the forwarding of an agent to an attacker-controlled host.
- CVE-2016-20012: OpenSSH through 8.7 allows remote attackers, who have a suspicion that a certain combination of username and public key is known to an SSH server, to test whether this suspicion is correct. This occurs because a challenge is sent only when that combination could be valid for a login session.

These vulnerabilities, following the discovery of exploits to exploit them, can have serious consequences for system security, including privilege escalation and the risk of unauthorized command execution.

The presence of the Proxmox web interface implies the existence of the active HTTPS service with its associated open port. This allowed for mapping between the IP address and the associated domain name. The domain name was obtained by inspecting the certificate within the web page reachable via the IP address and a specific port. This operation facilitated the effectiveness of subsequent scans by enabling the proper identification of resources within the machine.

To perform this operation, it was necessary to insert the corresponding association into the /etc/hosts file.

Another activity aimed at simplifying the use of subsequent commands related to the Proxmox virtual environment was to introduce an environment variable within Kali Linux.

This consisted of mapping a keyword to the domain name within the /etc/environment file. This way, it was possible to refer to the environment in question using the keyword preceded by the "$" character, indicating the selected environment variable.

The use of environment variables greatly simplifies interaction with the Proxmox environment, as it eliminates the need to manually type the IP address or the complete domain name each time commands or scans are executed.

The open port related to the use of the HTTPS service represents a critical component for system administration and management. However, due to the sensitivity and importance of this service, it is crucial to perform targeted analyses to ensure its security. Additionally, web applications are commonly vulnerable, so it is essential to conduct an in-depth analysis to detect potential weaknesses or exposures.

To conduct a targeted analysis of Proxmox's web applications, the Dirb tool, previously described, was used. Dirb is a useful tool for finding hidden or unauthorized directories within a web application. These directories may represent potential vulnerabilities or may contain sensitive information that should be protected.

However, in the analysis conducted, the output generated by Dirb did not return any significant results.

To further analyze files and directories, another test was conducted: fuzzing. Fuzzing aims to recheck the presence of hidden elements or unauthorized resources within the web application. However, in this case, the FFuF tool, previously described, was used in combination with some of the wordlists available within SecLists, which were downloaded and used as a basis during the preparation of the Kali Linux environment.

Despite using this advanced approach and combining tools and resources for fuzzing, the analysis did not yield significant results or detect hidden elements within the Proxmox web application.

In order to detect possible installations of WordPress, the WPScan tool, described earlier, was employed. However, as expected, no positive findings were obtained.

The last tool used in the analysis of the Proxmox environment's security was Nikto, described earlier. Nikto is a widely used tool for identifying potential vulnerabilities and configuration issues within web applications and servers. However, in this specific context, even the use of Nikto did not yield significant results.

4.2.3.2. Exploitation

Following the vulnerability analysis phase, an attempt was made to find a way to gain access to the machine. It's crucial to note that these operations were carried out assuming no knowledge of any access credentials.

Specifically, various avenues were pursued with the goal of gaining access to the system, either by obtaining credentials or by creating a reverse shell impersonating any user.

Initially, brute force attacks were executed against the SSH service and the Proxmox web interface login page. In both cases, the Hydra tool, described earlier, was used in combination with some of the wordlists available in SecLists and others created manually. However, both approaches did not yield positive results, indicating that the passwords used were not among those commonly found in the lists.

Another approach to gaining access to the system involved searching for exploits that could be used for the vulnerabilities identified in the previous phase. Despite multiple attempts, no exploits were found, and access was not successful.

Considering a scenario in which new exploits were developed for the specific context, it was chosen to simulate what could happen if an attacker managed to gain access as a non-privileged user.

Once inside the machine, it was possible to navigate through the directories to find potentially exploitable information. To facilitate this process, the LinPEAS tool, described earlier, was used, allowing for a comprehensive scan of the system, collecting a vast amount of information about configuration, running processes, services, system files, and permissions.

Specifically, users with administrator privileges were identified to attempt targeted brute force techniques again, but they were unsuccessful. Subsequently, a list of possible executable files with administrator privileges, those belonging to an administrator and with the Set User ID (SUID) bit assigned, was provided. However, no significant results were obtained.

Other relevant information provided by the tool included the identification of the Linux Kernel versions (5.15.102), and the Sudo utility version (1.9.5p2). To complete the identification of versions of relevant system components, specific commands were executed, revealing those of Proxmox version (7.4-3), and Debian version (11.6).

Based on the information just obtained, in-depth searches were conducted to identify potential vulnerabilities and any exploits that could be leveraged.

Specifically, for the Linux Kernel version, 240 vulnerabilities were identified. The most significant, based on known uses and potential damages, include:

- CVE-2023-0266: A use-after-free vulnerability exists in the ALSA PCM package in the Linux Kernel. SNDRV_CTL_IOCTL_ELEM_{READ|WRITE}32 is missing locks that can be used in a use-after-free that can result in privilege escalation to gain ring0 access from the system user.
- CVE-2023-32254: A flaw was found in the Linux kernel's ksmbd, a high-performance in-kernel SMB server. The specific flaw exists within the processing of SMB2_TREE_DISCONNECT commands. The issue results from the lack of proper locking when performing operations on an object. An attacker can leverage this vulnerability to execute code in the context of the kernel.
- CVE-2023-32250: A flaw was found in the Linux kernel's ksmbd, a high-performance in-kernel SMB server. The specific flaw exists within the processing of SMB2_SESSION_SETUP commands. The issue results from the lack of proper locking when performing operations on an object. An attacker can leverage this vulnerability to execute code in the context of the kernel.
- CVE-2022-2196: A regression exists in the Linux Kernel within KVM: nVMX that allowed for speculative execution attacks. L2 can carry out Spectre v2 attacks on L1 due to L1 thinking it doesn't need retpolines or IBPB after running L2 due to KVM (L0) advertising eIBRS support to L1. An attacker at L2 with code execution can execute code on an indirect branch on the host machine.
- CVE-2022-1012: A memory leak problem was found in the TCP source port generation algorithm in net/ipv4/tcp.c due to the small table perturb size. This flaw may allow an attacker to information leak and may cause a denial of service problem.

It's important to note that none of these vulnerabilities have exploitable exploits available.

For the Sudo version, the following vulnerabilities were identified:

- CVE-2023-22809: In Sudo before 1.9.12p2, the sudoedit (aka -e) feature mishandles extra arguments passed in the user-provided environment variables (SUDO_EDITOR, VISUAL, and EDITOR), allowing a local attacker to append arbitrary entries to the list of files to process. This can lead to privilege escalation. Affected versions are 1.8.0 through 1.9.12.p1. The problem exists because a user-specified editor may contain a "--" argument that defeats a protection mechanism, e.g., an EDITOR='vim -- /path/to/extra/file' value.

- CVE-2022-43995: Sudo 1.8.0 through 1.9.12, with the crypt() password backend, contains a plugins/sudoers/auth/passwd.c array-out-of-bounds error that can result in a heap-based buffer over-read. This can be triggered by arbitrary local users with access to Sudo by entering a password of seven characters or fewer. The impact could vary depending on the system libraries, compiler, and processor architecture.
- CVE-2023-28486: Sudo before 1.9.13 does not escape control characters in log messages.
- CVE-2023-28487: Sudo before 1.9.13 does not escape control characters in sudoreplay output.

Specifically, the CVE-2023-22809 has an exploitable exploit, but in the examined case, it didn't produce any results due to the absence of the required vim editor.

Finally, for both the Proxmox and Debian versions, no vulnerabilities were identified.

To conclude this phase and obtain a more comprehensive understanding of weaknesses, after gaining access as an administrator, the configuration files present in the system were analyzed. In particular, by examining the SSH configuration files, insecure policies were identified. Access was allowed from any IP address and for any user, including root, by entering the correct password. Furthermore, the last weakness identified was the absence of an active firewall.

4.2.3.3. Outcomes evaluation

Within this phase, a summary and analysis of everything obtained in the previous phases have been carried out. This allows for a complete overview of the system's situation with greater accessibility to the identified weaknesses and vulnerabilities.

Specifically, the main weaknesses pertain to the following incorrect configurations:

- SSH: It lacks a secure access methodology through key exchange and allows impersonation of any user, including root. Furthermore, since during the exploitation phase, brute force attacks were possible using the Hydra tool, it indicates the absence of defensive measures in this regard.
- Firewall: Its absence represents a weakness because all incoming and outgoing traffic is allowed without any limitations.
- Web interface login: Brute force attacks were possible through the Hydra tool.

Regarding vulnerabilities, they have been grouped into a table to display relevant information for each of them. Specifically, the table includes the following columns:

- Analyzed CVE
- Vulnerability source
- Potential impact: In this case, when possible, the associated CVSS (Common Vulnerability Scoring System) score has been used.
- Exploitation likelihood: In this case, when possible, the associated EPSS (Exploit Prediction Scoring System) score has been used.
- Expected consequence

CVE	Vulnerability source	Potential impact (CVSS)	Exploitation likelihood (EPSS)	Expected consequence
CVE-2021-41617	OpenSSH 8.4p1	7.0	0.06%	Privilege escalation
CVE-2021-36368	OpenSSH 8.4p1	3.7	0.19%	Privilege escalation
CVE-2021-28041	OpenSSH 8.4p1	7.1	0.17%	Code execution
CVE-2016-20012	OpenSSH 8.4p1	5.3	0.37%	Code execution
CVE-2023-0266	Linux Kernel 5.15.102	7.9	0.08%	Code execution
CVE-2023-32254	Linux Kernel 5.15.102	9.8	0.22%	Code execution
CVE-2023-32250	Linux Kernel 5.15.102	9.0	0.22%	Code execution
CVE-2022-2196	Linux Kernel 5.15.102	8.8	0.04%	Code execution
CVE-2022-1012	Linux Kernel 5.15.102	8.2	0.09%	Denial of service
CVE-2023-22809	Sudo 1.9.5p2	7.8	0.05%	Privilege escalation
CVE-2022-43995	Sudo 1.9.5p2	7.1	0.04%	Code execution
CVE-2023-28486	Sudo 1.9.5p2	5.3	0.05%	Code execution

CVE-2023-28487	Sudo 1.9.5p2	5.3	0.05%	Code execution

Table 1. Proxmox vulnerability report

A fundamental aspect for assessing the results involved risk analysis. This activity allows for the qualitative definition of the risk associated with each vulnerability to establish priorities in their management within the system. A common approach involves using tables that represent the probability of a threat occurring and the potential impact if it were to occur.

Specifically, for the table concerning the impact of vulnerabilities on the system, five levels were used, reflecting those provided by NIST regarding the CVSS.

Severity	**Impact**
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10

Table 2. Proxmox impact severity

Similarly, for the table regarding the likelihood of exploiting vulnerabilities, a table very similar to the previous one was used. In fact, once again, five levels and similar ranges were employed.

Severity	**Likelihood (%)**
Impossible	0.00
Less probable	0.01 - 39.99
Probable	40 - 69.99
High probable	70 - 89.99
Sure	90 - 100

Table 3. Proxmox likelihood severity

The combination of the tables just described resulted in the risk matrix, which serves as a guide for assigning the appropriate level to each vulnerability. Specifically, the impact values were placed in the columns, and the probability values in the rows.

	None	**Low**	**Medium**	**High**	**Critical**
Impossible	Inexistent	Inexistent	Inexistent	Inexistent	Inexistent
Less probable	Inexistent	Slight	Slight	Normal	Great
Probable	Inexistent	Slight	Normal	Great	Catastrophic
High probable	Inexistent	Normal	Great	Catastrophic	Catastrophic
Sure	Inexistent	Great	Catastrophic	Catastrophic	Catastrophic

Table 4. Proxmox risk matrix

After determining the tables to use for conducting the risk analysis, severity values for impact and probability were assigned to all the CVEs under examination.

CVE	**Impact**	**Likelihood**
CVE-2021-41617	High	Less probable
CVE-2021-36368	Low	Less probable
CVE-2021-28041	High	Less probable
CVE-2016-20012	Medium	Less probable
CVE-2023-0266	High	Less probable
CVE-2023-32254	Critical	Less probable
CVE-2023-32250	Critical	Less probable
CVE-2022-2196	High	Less probable
CVE-2022-1012	High	Less probable
CVE-2023-22809	High	Less probable
CVE-2022-43995	High	Less probable
CVE-2023-28486	Medium	Less probable
CVE-2023-28487	Medium	Less probable

Table 5. Proxmox vulnerability severities

Finally, the associated risk for each CVE was determined.

CVE	Risk
CVE-2021-41617	Normal
CVE-2021-36368	Slight
CVE-2021-28041	Normal
CVE-2016-20012	Slight
CVE-2023-0266	Normal
CVE-2023-32254	Great
CVE-2023-32250	Great
CVE-2022-2196	Normal
CVE-2022-1012	Normal
CVE-2023-22809	Normal
CVE-2022-43995	Normal
CVE-2023-28486	Slight
CVE-2023-28487	Slight

Table 6. Proxmox vulnerability risks

Thirteen vulnerabilities were analyzed. It is worth noting that the risk analysis reported medium to low risk values for most of the identified vulnerabilities. However, there are two cases of great risk related to the Linux Kernel.

The last activity carried out in this phase consists of generating the RAV, which is the comprehensive and structured document that provides an overall representation of the system's attack surface. To obtain the RAV associated with the Proxmox environment, the Excel spreadsheet provided by ISECOM was used.

The compilation of the RAV involved following the guidelines provided in the previous chapter. In particular, for the Visibility category, the accounts present within the system were counted, and for the Access category, all the open ports identified from the previous scans were used.

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	
Visibility	2
Access	4
Trust	0
Total (Porosity)	6

OPSEC
7,722143

CONTROLS

Class A		Missing
Authentication	2	4
Indemnification	0	6
Resilience	4	2
Subjugation	0	6
Continuity	0	6
Total Class A	6	24

True Controls
5,097113

Full Controls
5,097113

True Coverage A
20,00%

Class B		Missing
Non-Repudiation	4	2
Confidentiality	2	4
Privacy	0	6
Integrity	2	4
Alarm	4	2
Total Class B	12	18

True Coverage B
40,00%

Total True Coverage
30,00%

		True Missing
All Controls Total	18	42
Whole Coverage	30,00%	70,00%

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM
www.osstmm.org

LIMITATIONS		Item Value	Total Value
Vulnerabilities	248	8,000000	1984,000000
Weaknesses	3	5,000000	15,000000
Concerns	0	4,000000	0,000000
Exposures	0	42,533333	0,000000
Anomalies	0	41,833333	0,000000
Total # Limitations	251		1999,0000

Limitations
28,098639

Security Δ
-30,72

True Protection
69,28

Actual Security: 69,6203 ravs

Figure 11. Proxmox RAV

4.2.3.4. Hardening

Regarding the identification of weaknesses and vulnerabilities in the previous phases, it is essential to identify and apply the best possible hardening practices.

The first and most important step to significantly reduce the number of vulnerabilities affecting the system is to perform all available updates. This ensures that the kernel, operating system, and tools are upgraded to the latest versions in line with the most recent security measures provided by developers.

Specifically, the elements that most require updates are the Linux Kernel, OpenSSH, and Sudo. To do this, it may be sufficient to update Proxmox and the installed packages.

The most effective hardening for the SSH service involves its complete deactivation. However, if it is considered a vital resource, it is advisable to intervene in its configuration file, i.e., /etc/ssh/sshd_config.

In particular, the following operations should be carried out:

- Disabling password authentication: This procedure enforces the use of keys (public key, private key) to mitigate brute force attempts. To apply this change, you need to set the "PasswordAuthentication no" parameter. However, if you wish to keep password authentication, it would be advisable to prevent empty passwords using the "PermitEmptyPasswords no" parameter.
- Use of additional tools: To mitigate brute force attacks, you can use tools like Fail2Ban. It monitors failed login attempts from an IP address, and if the number of failed attempts exceeds a certain threshold within a specific time interval, that address is banned for a certain period.
- Disabling root user login: The root user should not be active in a system, as the current standard is to use administrator permissions through sudoers. Therefore, it is advisable to disable root access using the "PermitRootLogin no" parameter.
- Changing the default or common SSH port: Most exploits target common SSH ports. Changing the port can reduce the potential number of attacks. You can apply this modification using the "Port ###" parameter.
- Enabling inactivity timeout: An SSH connection can remain active without any activity, posing a security risk. It is a good practice to configure an inactivity timeout interval. To do so, you need to set the "ClientAliveInterval ###" parameter.
- Allowing access only to specific users or groups using the "AllowUsers" and/or "AllowGroups" parameters.

- Allowing access only from specific IP addresses using the "ListenAddress" parameter.

Another hardening policy to implement is the implementation of a firewall. Proxmox provides multiple ways to do this, but the most convenient and intuitive method is to apply filtering rules directly from the web interface.

Specifically, you can implement the firewall comprehensively, both on the server and on the Proxmox machine itself. Additionally, you can modify rules related to virtual machines within the virtualization environment from the web interface. However, since the virtual machines are copies of real systems, it is more appropriate to intervene directly within those virtual machines in subsequent analyses.

In the context of the Proxmox system, recommended rules involve rejecting all incoming and outgoing packets and then only accepting those necessary for the operation of essential services. This way, an allow list policy is implemented, representing one of the best practices in cybersecurity.

Another hardening practice is to prevent brute force attacks, protecting accounts from unauthorized access by using PAM Faillock. Faillock is a PAM module that keeps track of failed user login attempts and manages the blocking of accounts once a certain threshold is exceeded.

One final hardening measure to enhance security is to enforce the use of strong passwords for system users. As indicated by the lack of success in brute force attempts on both the Proxmox web interface login page and the SSH service, the current passwords appear to be complex enough, but there is no control method in place to enforce this choice. Therefore, you can use the libpam-pwquality library, which, through a configuration file, allows you to specify the minimum requirements for each password. Finally, another good password practice is to set an expiration time to ensure they are changed periodically.

4.2.4. Virtual machine: Robotcup

4.2.4.1. Vulnerability analysis

As in the previous case, to assess the security of the virtual machine named Robotcup, various scans were performed using the Kali Linux VM.

The first task involved identifying open ports. Once again, the Nmap tool was used for this analysis, performing both types of scans described earlier. The scans did not reveal open ports other than those used for HTTP and HTTPS protocols. Typing the IP address into a web browser allowed access to the website provided by the machine. Similarly, the mapping between the IP address and associated domain name was established within the /etc/hosts file. The domain name was obtained through the examination of the certificate found within the accessed web page. Additionally, the mapping between the keyword to be used as an environment variable and the domain name was created within the /etc/environment file.

The open ports identified represent a critical component for system administration and management. Therefore, it is essential to carry out targeted analyses to detect potential weaknesses or exposures.

As in the previous case, the Dirb and FFuF tools were used to find hidden or unauthorized files and directories. The most significant result was the discovery of the login page dedicated to the WordPress service. WordPress is widely used open-source software for creating and managing websites. It is known for its user-friendliness and flexibility, allowing users to publish content online without requiring advanced knowledge of programming or web design. Additionally, WordPress offers a wide range of themes and plugins for customizing the appearance and functionality of websites to suit individual needs. However, its presence is noteworthy because, in the field of cybersecurity, this service is known to be susceptible to multiple threats.

Other information obtained from these scans included the identification of a login page for the machine's website, the presence of files and directories for WordPress use that could be explored by anyone, and the presence of the XML-RPC protocol configuration file. XML-RPC is a communication protocol that allows third parties to interact with WordPress sites. Enabling XML-RPC can pose some security risks; as it is an outdated protocol, it may be susceptible to attacks such as password brute force or Distributed Denial of Service (DDoS) attacks.

Since WordPress was identified, the WPScan tool was employed to detect potential vulnerabilities and weaknesses. The information obtained was of great importance because it revealed:

- The service version, which was 6.2.2, representing the most up-to-date version possible.
- The presence of the "Square" theme characterized by an outdated version.
- Registered users on the platform.
- The active XML-RPC protocol, confirming the previous findings.
- The absence of known vulnerabilities and exploits for the service's version.

The last tool used in the security analysis was Nikto, which aimed to identify potential vulnerabilities and configuration issues within web applications. However, even in this case, no significant results were obtained.

4.2.4.2. Exploitation

Following the vulnerability analysis phase, efforts were made to find a way to access the machine.

Similar to the previous case, various approaches were pursued with the goal of gaining access to the system, obtaining credentials, or creating a reverse shell impersonating any user.

Initially, as awareness of users with access to the WordPress platform existed, brute force attacks were conducted on the login page using the Hydra tool. However, no positive results were returned, indicating that the passwords used did not match those found in common password lists.

Instead, during attempts to use users not listed in the WPScan tool's scan, an Information Leakage weakness was revealed. An error message was displayed based on the existence of the user attempting to log in.

Subsequently, brute force attacks were also performed on the login page of the website provided by the machine and on the XML-RPC protocol, again using the Hydra tool. In both cases, no significant results were obtained.

Extensive research was conducted to identify possible exploits related to the "Square" theme reported by the WPScan tool, but this search was unsuccessful.

Considering a scenario in which new exploits have been developed for the specific context, it was simulated what could happen if an attacker managed to access the system as a non-privileged user.

To achieve this, since access to the machine could not be gained directly, demonstrating a reasonable level of security, an action was taken that leverages the administrator privileges held for the Proxmox virtualization environment. Specifically, a forced reboot in Recovery Mode was performed on the VM under examination. This allowed the setting of an arbitrary password for the root user to ensure access at any time and the creation of a new non-privileged user.

As in the previous case, once inside the machine, it was possible to navigate through directories to identify potentially exploitable information. To facilitate this process, the LinPEAS tool was used, allowing for a comprehensive system scan, collecting a vast amount of information about the configuration, running processes, services, system files, and permissions.

Specifically, administrators were identified to carry out targeted brute force techniques, but they were unsuccessful.

Subsequently, a list of possible executable files with administrator privileges, i.e., those belonging to an administrator and with the SUID bit assigned, was provided. However, no significant results were obtained.

Other relevant information provided by the tool included the identification of the Linux Kernel version (5.4.0), the Ubuntu operating system version (20.04.6), the Sudo utility version (1.8.31), the MySQL database version (8.0.33). To complete the identification of versions of relevant system components, specific commands were executed, returning the Apache version (2.4.41).

Based on the information obtained, in-depth searches were conducted to identify potential vulnerabilities and any exploits that could be used.

Specifically, for the Linux Kernel version, 548 vulnerabilities were identified. The most significant ones, based on known uses and potential damage, included:

- CVE-2021-22555: A heap out-of-bounds write affecting Linux since v2.6.19-rc1 was discovered in net/netfilter/x_tables.c. This allows an attacker to gain privileges or cause a DoS (via heap memory corruption) through user namespace.
- CVE-2023-0266: A use-after-free vulnerability exists in the ALSA PCM package in the Linux Kernel.
- CVE-2022-34918: An issue was discovered in the Linux kernel through 5.18.9. A type confusion bug in nft_set_elem_init (leading to a buffer overflow) could be used by a local attacker to escalate privileges.

- CVE-2022-0435: A stack overflow flaw was found in the Linux kernel's TIPC protocol functionality.
- CVE-2020-12351: Improper input validation in BlueZ may allow an unauthenticated user to potentially enable escalation of privilege via adjacent access.

It's important to note that CVE-2021-22555 and CVE-2022-34918 had exploitable exploits, but they did not yield any results due to the lack of the required gcc and make tools.

For the Sudo version, the following vulnerabilities were identified:

- CVE-2023-28487: Sudo before 1.9.13 does not escape control characters in sudoreplay output.
- CVE-2023-28486: Sudo before 1.9.13 does not escape control characters in log messages.
- CVE-2023-22809: In Sudo before 1.9.12p2, the sudoedit (aka -e) feature mishandles extra arguments passed in the user-provided environment variables.
- CVE-2022-43995: Sudo 1.8.0 through 1.9.12, with the crypt() password backend, contains a plugins/sudoers/auth/passwd.c array-out-of-bounds error.
- CVE-2021-23240: Selinux_edit_copy_tfiles in sudoedit in Sudo before 1.9.5 allows a local unprivileged user to gain file ownership and escalate privileges.
- CVE-2021-23239: The sudoedit personality of Sudo before 1.9.5 may allow a local unprivileged user to perform arbitrary directory-existence tests by winning a sudo_edit.c race condition.
- CVE-2021-3156: Sudo before 1.9.5p2 contains an off-by-one error that can result in a heap-based buffer overflow.

It's important to note that CVE-2023-22809 and CVE-2021-3156 had exploitable exploits but did not produce any results. Specifically, CVE-2021-3156 was hindered by the absence of the required make tool.

For the Apache versions, 35 vulnerabilities were identified. The most significant ones included:

- CVE-2020-11984: Apache HTTP server 2.4.32 to 2.4.44 mod_proxy_uwsgi info disclosure and possible RCE.
- CVE-2021-26691: In Apache HTTP Server versions 2.4.0 to 2.4.46, a specially crafted SessionHeader sent by an origin server could cause a heap overflow.

- CVE-2021-39275: ap_escape_quotes() may write beyond the end of a buffer when given malicious input.
- CVE-2021-44790: A carefully crafted request body can cause a buffer overflow in the mod_lua multipart parser.
- CVE-2022-23943: Out-of-bounds Write vulnerability in mod_sed of Apache HTTP Server allows an attacker to overwrite heap memory with possibly attacker provided data.

It's important to note that none of these vulnerabilities had exploitable exploits.

Finally, for the Ubuntu and MySQL versions, no vulnerabilities were identified.

The last critical information provided by LinPEAS concerned the identification of two separate files containing sensitive data for accessing two databases, one related to the web service provided by the VM in use and one related to WordPress.

Indeed, using these credentials, it was possible to explore the databases thoroughly to identify further sensitive data by performing the necessary SQL queries. Specifically, it was possible to retrieve all username and password associations of all registered users. It's worth noting that the passwords were not obtained in plain text, as they were properly hashed within the database. However, starting from these associations, password cracking attempts were made to deduce the original passwords using the Hydra tool, but no positive results were obtained.

To conclude this phase and achieve greater completeness in identifying weaknesses, after gaining administrator access, the configuration files present in the system were analyzed.

In particular, this analysis revealed the presence of suboptimal configurations regarding the firewall and VM boot policies. In fact, the latter allowed the machine to be rebooted in Recovery Mode.

4.2.4.3. Outcomes evaluation

Within this phase, a summary and analysis of everything obtained in the previous phases have been carried out. This way, it is possible to have a comprehensive overview of the situation within the system with better accessibility to the identified weaknesses and vulnerabilities.

Specifically, the main weaknesses concern the following incorrect configurations:

- Machine boot: There are no obstacles to rebooting the machine, ensuring it can start in Recovery Mode.
- Firewall: All outgoing packets are allowed, the port related to HTTP is unnecessary, and the generated logs have a low level of detail.
- Sensitive files: Anyone is granted read permissions for files containing MySQL database access credentials.
- Web service login: Brute force attacks were possible.
- WordPress login: Distinct messages were provided for users registered with the service and those who are not. Additionally, brute force attacks were possible.
- Directory traversal: It was possible to navigate within some folders and files of WordPress.
- XML-RPC: This protocol allows for the execution of brute force attacks.

As for vulnerabilities, similar to the previous case, they have been grouped into a table to display relevant information for each of them.

CVE	Vulnerability source	Potential impact (CVSS)	Exploitation likelihood (EPSS)	Expected consequence
CVE-2021-22555	Linux Kernel 5.4.0	8.3	0.26%	Code execution & Denial of service
CVE-2023-0266	Linux Kernel 5.4.0	7.9	0.08%	Code execution
CVE-2022-34918	Linux Kernel 5.4.0	8.3	0.26%	Code execution
CVE-2022-0435	Linux Kernel 5.4.0	9.0	1.00%	Code execution

CVE-2020-12351	Linux Kernel 5.4.0	8.8	0.16%	Code execution
CVE-2023-28487	Sudo 1.8.31	5.3	0.05%	Code execution
CVE-2023-28486	Sudo 1.8.31	5.3	0.05%	Code execution
CVE-2023-22809	Sudo 1.8.31	7.8	0.05%	Privilege escalation
CVE-2022-43995	Sudo 1.8.31	7.1	0.04%	Code execution
CVE-2021-23240	Sudo 1.8.31	7.8	0.08%	Code execution
CVE-2021-23239	Sudo 1.8.31	2.5	0.05%	Code execution
CVE-2021-3156	Sudo 1.8.31	7.8	96.79%	Code execution & Privilege escalation
CVE-2020-11984	Apache 2.4.41	9.8	1.08%	Code execution
CVE-2021-26691	Apache 2.4.41	9.8	94.51%	Code execution
CVE-2021-39275	Apache 2.4.41	9.8	3.74%	Code execution
CVE-2021-44790	Apache 2.4.41	9.8	16.91%	Code execution
CVE-2022-23943	Apache 2.4.41	9.8	80.53%	Code execution

Table 7. Robotcup vulnerability report

A fundamental aspect of evaluating the results involved risk analysis. This activity allows for the qualitative definition of the risk associated with each vulnerability to establish a prioritization in their management within the system. A common approach involves the use of tables that represent the probability of a threat occurring and the potential impact if it were to occur.

Specifically, the tables used were the same as in the previous case.

Severity	Impact
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10

Table 8. Robotcup impact severity

Severity	Likelihood (%)
Impossible	0.00
Less probable	0.01 - 39.99
Probable	40 - 69.99
High probable	70 - 89.99
Sure	90 - 100

Table 9. Robotcup likelihood severity

	None	Low	Medium	High	Critical
Impossible	Inexistent	Inexistent	Inexistent	Inexistent	Inexistent
Less probable	Inexistent	Slight	Slight	Normal	Great
Probable	Inexistent	Slight	Normal	Great	Catastrophic
High probable	Inexistent	Normal	Great	Catastrophic	Catastrophic
Sure	Inexistent	Great	Catastrophic	Catastrophic	Catastrophic

Table 10. Robotcup risk matrix

After determining the tables to use for conducting the risk analysis, severity values for impact and probability were assigned to all the CVEs under examination.

CVE	Impact	Likelihood
CVE-2021-22555	High	Less probable

CVE-2023-0266	High	Less probable
CVE-2022-34918	High	Less probable
CVE-2022-0435	Critical	Less probable
CVE-2020-12351	High	Less probable
CVE-2023-28487	Medium	Less probable
CVE-2023-28486	Medium	Less probable
CVE-2023-22809	High	Less probable
CVE-2022-43995	High	Less probable
CVE-2021-23240	High	Less probable
CVE-2021-23239	Low	Less probable
CVE-2021-3156	High	Sure
CVE-2020-11984	Critical	Less probable
CVE-2021-26691	Critical	Sure
CVE-2021-39275	Critical	Less probable
CVE-2021-44790	Critical	Less probable
CVE-2022-23943	Critical	High probable

Table 11. Robotcup vulnerability severities

Finally, the associated risk for each CVE was determined.

CVE	Risk
CVE-2021-22555	Normal
CVE-2023-0266	Normal
CVE-2022-34918	Normal
CVE-2022-0435	Great
CVE-2020-12351	Normal
CVE-2023-28487	Slight
CVE-2023-28486	Slight
CVE-2023-22809	Normal

CVE-2022-43995	Normal
CVE-2021-23240	Normal
CVE-2021-23239	Slight
CVE-2021-3156	Catastrophic
CVE-2020-11984	Great
CVE-2021-26691	Catastrophic
CVE-2021-39275	Great
CVE-2021-44790	Great
CVE-2022-23943	Catastrophic

Table 12. Robotcup vulnerability risks

Seventeen vulnerabilities were analyzed. Note that the risk analysis reported medium to high-risk values for most of the identified vulnerabilities. Additionally, there are three cases of catastrophic risk related to Sudo and Apache.

The last activity carried out within this phase consists of generating the RAV, which is a comprehensive and structured document providing an overall representation of the system's attack surface. To obtain the RAV associated with the Robotcup machine, the Excel spreadsheet provided by ISECOM was utilized.

The compilation of the RAV involved following the guidelines provided in the previous chapter. Specifically, for the Visibility category, all the accounts present within the system were counted, and for the Access category, all the open ports identified in the previous scans were used.

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	
Visibility	4
Access	2
Trust	0
Total (Porosity)	6

OPSEC 7,722143

CONTROLS		
Class A		Missing
Authentication	2	4
Indemnification	0	6
Resilience	2	4
Subjugation	0	6
Continuity	0	6
Total Class A	4	26

True Controls 4,017304

Full Controls 4,017304

True Coverage A 13,33%

Class B		Missing
Non-Repudiation	2	4
Confidentiality	1	5
Privacy	0	6
Integrity	1	5
Alarm	2	4
Total Class B	6	24

True Coverage B 20,00%

		True Missing
All Controls Total	10	50
Whole Coverage	16,67%	83,33%

Total True Coverage 16,67%

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM www.osstmm.org

LIMITATIONS		Item Value	Total Value
Vulnerabilities	590	9,333333	5506,666667
Weaknesses	7	5,333333	37,333333
Concerns	0	5,000000	0,000000
Exposures	0	100,333333	0,000000
Anomalies	0	99,500000	0,000000
Total # Limitations	597		5544,0000

Limitations 32,991514

Security Δ -36,70

True Protection 63,30

Actual Security: 64,2157 ravs

Figure 12. Robotcup RAV

4.2.4.4. Hardening

Following the identification of weaknesses and vulnerabilities and their analysis in the previous phases, it is crucial to identify and apply the best possible hardening practices.

The first and most important operation to significantly reduce the number of vulnerabilities affecting the system is to perform all available updates. This ensures the advancement of the kernel, operating system, and tools to the latest versions in line with the most up-to-date security measures provided by developers. Specifically, the elements that require updates are the Linux Kernel and Sudo. To do this, updating Ubuntu and the installed packages might be sufficient.

The most effective hardening measure to prevent an attacker from restarting the VM in Recovery Mode to enter or modify the root user's password is to set a boot policy that requires the administrator's password. This can be achieved by placing an appropriate configuration file inside the /etc/grub.d directory. It's important to note that files in this directory are typically readable by any user of the system. Therefore, it's necessary to modify these permissions so that only the administrator can read the file and enter the password as a hashed password.

Another hardening policy to implement is modifying the firewall configuration, which is already quite secure. In particular, the firewall is constructed using Uncomplicated Firewall (UFW), a high-level and user-oriented firewall that leverages the power of IPTables, a low-level firewall, while simplifying its use through a more intuitive interface and commands. The initial changes involve increasing the level of detail in automatically generated log files and denying all outgoing packets, allowing only those from necessary services. Additionally, consider closing the port related to the HTTP protocol, leaving only the one for HTTPS, which, thanks to certificates, provides greater security.

For hardening the files that allowed obtaining access credentials for MySQL databases, it is crucial to change their permissions so that users other than the administrator cannot read or modify them.

Regarding WordPress, various hardening practices can be applied. Specifically, to prevent Information Leakage on the login page, which reveals whether the user used for login exists, you can insert specific lines of code into the functions.php file. Furthermore, you can use the following plugins to enhance the security of the service:

- Limit Login Attempts Reloaded: It can be configured to limit the number of failed password login attempts and block them for a specified period.
- Manage XML-RPC: It can be configured to verify incoming requests through the XML-RPC protocol to determine whether they should be accepted or denied. This can be done without using any plugins by modifying the /var/www/wordpress/.htaccess configuration file.
- Disable XML-RPC-API: Completely disables the XML-RPC protocol. Again, this can be done without using any plugins by modifying the /var/www/wordpress/.htaccess configuration file.

Additionally, it is necessary to prevent directory traversal attacks in WordPress. To do this, you need to insert specific lines of code at the end of the /var/www/wordpress/.htaccess file.

Another hardening practice is to prevent brute force attacks by protecting accounts from unauthorized access using PAM Faillock. Faillock is a PAM module that tracks failed login attempts by users and manages account locking after surpassing a certain threshold.

A final hardening measure to enhance security is to enforce the use of strong passwords for system users and those registered in databases. As indicated by the lack of success in brute force practices applied to the login page of the VM's web interface, as well as to WordPress, and the failure in cracking passwords within the databases, the currently used passwords are already quite complex. However, there is no enforcement method in place. To induce the use of strong passwords, you can utilize MySQL's validate_password component. This tool enforces a policy that users must follow when choosing their password.

For system users, you can use the libpam-pwquality library, which allows you to specify the minimum requirements for each password through a configuration file. Finally, another good practice regarding passwords is to set an expiration time so that they are changed periodically.

4.2.5. Virtual machine: Docenti

4.2.5.1. Vulnerability analysis

As in previous cases, various scans were conducted through the Kali Linux VM to assess the security of the virtual machine named Docenti.
The initial activity involved identifying open ports. Once again, the Nmap tool was used for this analysis, performing both types of scans described earlier.

From the scans, the most significant result, in addition to the open port used for the HTTP protocol, was the identification of an open port associated with the SSH service, revealing the use of OpenSSH version 6.6p1. This information is particularly relevant because this version of OpenSSH is known to be subject to 26 documented security vulnerabilities.
The most significant vulnerabilities, based on known exploits and potential damage, include:

- CVE-2016-1908: This vulnerability in OpenSSH before 7.2 allows remote X11 clients to gain trusted X11 forwarding privileges by triggering a fallback when there are configuration issues on the X11 server.
- CVE-2015-5600: This vulnerability in OpenSSH through 6.9 makes it easier for remote attackers to conduct brute-force attacks or cause a denial of service via a long list in the ssh -oKbdInteractiveDevices option.
- CVE-2016-0778: This vulnerability allows remote servers to cause a denial of service or possibly have other impacts by requesting many forwardings.
- CVE-2016-10012: This vulnerability in OpenSSH before 7.4 might allow local users to gain privileges.
- CVE-2016-10708: This vulnerability in OpenSSH before 7.4 allows remote attackers to cause a denial of service via an out-of-sequence NEWKEYS message.

These vulnerabilities, once exploits are discovered for their use, can have serious consequences for system security, such as privilege escalation and the risk of executing unauthorized commands.
After discovering the open port related to the HTTP protocol, entering the IP address in the web browser allowed access to the site provided by the machine in question. Once again, the /etc/hosts file was used to map the IP address to the associated domain name, which was obtained through browsing the reached web page. Furthermore, the mapping between

the keyword to be used as an environment variable and the domain name was created in the /etc/environment file.

The identified open ports represent a critical component for system administration and management. Therefore, it is essential to perform targeted analyses to detect potential weaknesses or exposures.

As in the previous case, the Dirb and FFuF tools were used to identify hidden or unauthorized files and directories. However, no significant results were obtained.

To detect the possible presence of WordPress installations, the WPScan tool was employed. However, as expected, no positive results were obtained.

The last tool used in the security analysis was Nikto, which is used to identify potential vulnerabilities and configuration issues within web applications. Again, no significant results were obtained in this case either.

4.2.5.2. Exploitation

Following the vulnerability analysis phase, efforts were made to find a way to access the machine.

As in previous cases, various approaches were pursued with the goal of gaining access to the system by obtaining credentials or creating a reverse shell impersonating any user.

Initially, brute force attacks were carried out against the SSH service. Hydra tool was used for this purpose. However, no positive results were returned, indicating that the passwords used do not match those found in common password lists.

Extensive research was conducted to identify potential exploits related to the vulnerabilities discovered in the previous phase. However, after multiple attempts, no exploits were found, and access was not successful.

Considering a scenario in which new exploits have been developed for the context at hand, it was chosen to simulate what could happen if an attacker managed to gain access as a non-privileged user.

To achieve this, since it was not possible to access the machine directly, a privileged action was taken to exploit the Proxmox virtualization environment's administrator privileges. Specifically, a forced reboot in Recovery Mode was performed for the VM in question. This allowed setting an arbitrary password for the root user to ensure access at any time and creating a new non-privileged user.

As in previous cases, once inside the machine, it was possible to navigate through folders to find potentially exploitable information. The LinPEAS tool was used to facilitate this

process, allowing a comprehensive scan of the system, collecting a wide range of information about configuration, running processes, services, system files, and permissions.

Specifically, administrators with privileges were identified to execute targeted brute force techniques again, but they were unsuccessful.

Subsequently, a list of possible executable files with administrator privileges, those belonging to an administrator with the SUID bit set, was provided. However, no significant results were obtained.

Other relevant information provided by the tool included the identification of the Linux Kernel version (3.13), the Ubuntu operating system version (14.04.6), the Sudo utility version (1.8.9p5), and the MySQL database version (14.14). To complete the identification of the versions of the relevant system components, specific commands were executed to retrieve the Apache version (2.4.7).

Based on the information obtained, extensive research was conducted to identify potential vulnerabilities and any exploits to exploit them.

Specifically, for the Linux Kernel version, 1322 vulnerabilities were identified. The most significant ones, based on known uses and potential damage, include:

- CVE-2016-5195: Race condition in mm/gup.c in the Linux kernel 2.x through 4.x before 4.8.3 allows local users to gain privileges by leveraging incorrect handling of a copy-on-write (COW) feature to write to a read-only memory mapping, as exploited in the wild in October 2016, aka "Dirty COW".
- CVE-2021-22555: A heap out-of-bounds write affecting Linux since v2.6.19-rc1 was discovered in net/netfilter/x_tables.c. This allows an attacker to gain privileges or cause a DoS (via heap memory corruption) through user name space
- CVE-2015-1328: The overlayfs implementation in the linux (aka Linux kernel) package before 3.19.0-21.21 in Ubuntu through 15.04 does not properly check permissions for file creation in the upper filesystem directory, which allows local users to obtain root access by leveraging a configuration in which overlayfs is permitted in an arbitrary mount namespace.
- CVE-2016-4997: The compat IPT_SO_SET_REPLACE and IP6T_SO_SET_REPLACE setsockopt implementations in the netfilter subsystem in the Linux kernel before 4.6.3 allow local users to gain privileges or cause a denial of service (memory corruption) by leveraging in-container root access to provide a crafted offset value that triggers an unintended decrement.

- CVE-2016-8655: Race condition in net/packet/af_packet.c in the Linux kernel through 4.8.12 allows local users to gain privileges or cause a denial of service (use-after-free) by leveraging the CAP_NET_RAW capability to change a socket version, related to the packet_set_ring and packet_setsockopt functions.

It's important to note that all the mentioned CVEs have exploitable exploits, especially CVE-2016-5195, which was successfully exploited for privilege escalation.

For the Sudo version, 15 vulnerabilities were identified. The most significant ones include:

- CVE-2021-3156: Sudo before 1.9.5p2 contains an off-by-one error that can result in a heap-based buffer overflow, which allows privilege escalation to root via "sudoedit -s" and a command-line argument that ends with a single backslash character.
- CVE-2023-22809: In Sudo before 1.9.12p2, the sudoedit (aka -e) feature mishandles extra arguments passed in the user-provided environment variables (SUDO_EDITOR, VISUAL, and EDITOR), allowing a local attacker to append arbitrary entries to the list of files to process. This can lead to privilege escalation. Affected versions are 1.8.0 through 1.9.12.p1. The problem exists because a user-specified editor may contain a "--" argument that defeats a protection mechanism.
- CVE-2019-14287: In Sudo before 1.8.28, an attacker with access to a Runas ALL sudoer account can bypass certain policy blacklists and session PAM modules, and can cause incorrect logging, by invoking sudo with a crafted user ID. For example, this allows bypass of !root configuration, and USER= logging, for a "sudo -u \#$((0xffffffff))" command.
- CVE-2016-7076: Sudo before version 1.8.18p1 is vulnerable to a bypass in the sudo noexec restriction if application run via sudo executed wordexp() C library function with a user supplied argument. A local user permitted to run such application via sudo with noexec restriction could possibly use this flaw to execute arbitrary commands with elevated privileges.
- CVE-2019-18634: In Sudo before 1.8.26, if pwfeedback is enabled in /etc/sudoers, users can trigger a stack-based buffer overflow in the privileged sudo process. (pwfeedback is a default setting in Linux Mint and elementary OS; however, it is NOT the default for upstream and many other packages, and would exist only if enabled by an administrator.) The attacker needs to deliver a long string to the stdin of getln() in tgetpass.c.

Notably, CVE-2023-22809 and CVE-2021-3156 have exploitable exploits, but they did not yield positive results in this case, with CVE-2021-3156 failing due to the absence of the required make tool.

For the Apache version, 63 vulnerabilities were identified. The most significant ones include:

- CVE-2017-3167: In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.
- CVE-2017-7679: In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.
- CVE-2018-1312: In Apache httpd 2.2.0 to 2.4.29, when generating an HTTP Digest authentication challenge, the nonce sent to prevent reply attacks was not correctly generated using a pseudo-random seed. In a cluster of servers using a common Digest authentication configuration, HTTP requests could be replayed across servers by an attacker without detection.
- CVE-2021-26691: In Apache HTTP Server versions 2.4.0 to 2.4.46 a specially crafted SessionHeader sent by an origin server could cause a heap overflow.
- CVE-2021-39275: ap_escape_quotes() may write beyond the end of a buffer when given malicious input. No included modules pass untrusted data to these functions, but third-party / external modules may. This issue affects Apache HTTP Server 2.4.48 and earlier.

None of these vulnerabilities were found to have exploitable exploits.

For the Ubuntu version, the following vulnerabilities were identified:

- CVE-2021-3492: Shiftfs, an out-of-tree stacking file system included in Ubuntu Linux kernels, did not properly handle faults occurring during copy_from_user() correctly. These could lead to either a double-free situation or memory not being freed at all. An attacker could use this to cause a denial of service (kernel memory exhaustion) or gain privileges via executing arbitrary code. AKA ZDI-CAN-13562.
- CVE-2021-3493: The overlayfs implementation in the linux kernel did not properly validate with respect to user namespaces the setting of file capabilities on files in an underlying file system. Due to the combination of unprivileged user namespaces along with a patch carried in the Ubuntu kernel to allow unprivileged overlay mounts, an attacker could use this to gain elevated privileges.

- CVE-2015-1328: The overlayfs implementation in the linux (aka Linux kernel) package before 3.19.0-21.21 in Ubuntu through 15.04 does not properly check permissions for file creation in the upper filesystem directory, which allows local users to obtain root access by leveraging a configuration in which overlayfs is permitted in an arbitrary mount namespace.

CVE-2021-3493 and CVE-2015-1328 have exploitable exploits, but they did not yield positive results in this case.

Finally, for the MySQL version, no vulnerabilities were identified.

The last critical piece of information provided by LinPEAS was the discovery of a file containing sensitive data for accessing the web service database provided by the VM. Using these credentials, it was possible to explore the database fully, attempting to retrieve further sensitive data through SQL queries. It's important to note that the passwords were not obtained in plaintext since they were stored securely in the database. However, attempts to crack the passwords using Hydra based on the obtained username-password associations did not yield positive results.

To conclude this phase and gain a more comprehensive understanding of the weaknesses, an analysis of the configuration files present in the system was conducted. In particular, this analysis revealed suboptimal configurations related to SSH, the firewall, and the VM's boot policies. Specifically, the VM allowed rebooting in Recovery Mode, potentially posing a security risk.

4.2.5.3. Outcomes evaluation

Within this phase, a summary and analysis of everything obtained in the previous phases have been conducted. This approach allows for a comprehensive overview of the system's current state, providing better accessibility to the identified weaknesses and vulnerabilities.

Specifically, the main weaknesses pertain to the following incorrect configurations:

- Machine Boot: There are no obstacles to restarting the machine, ensuring booting into Recovery Mode.
- Firewall: All outgoing packets are allowed, and the generated logs have a low level of detail.
- Sensitive Files: Read permissions are granted to anyone for the file containing MySQL database access credentials.
- SSH: Examining the SSH configuration file, insecure policies were identified. Additionally, brute force attacks were possible.

As for vulnerabilities, similar to the previous case, they have been grouped within a table to display relevant information for each of them.

CVE	Vulnerability source	Potential impact (CVSS)	Exploitation likelihood (EPSS)	Expected consequence
CVE-2016-1908	OpenSSH 6.6p1	9.8	0.37%	Privilege escalation
CVE-2015-5600	OpenSSH 6.6p1	8.5	19.08%	Denial of service
CVE-2016-0778	OpenSSH 6.6p1	8.1	0.33%	Code execution & Denial of service
CVE-2016-10012	OpenSSH 6.6p1	7.8	0.04%	Privilege escalation
CVE-2016-10708	OpenSSH 6.6p1	7.5	4.68%	Denial of service
CVE-2016-5195	Linux Kernel 3.13	7.8	87.94%	Privilege escalation
CVE-2021-22555	Linux Kernel 3.13	8.3	0.26%	Code execution & Denial of service
CVE-2015-1328	Linux Kernel 3.13	7.8	0.06%	Privilege escalation
CVE-2016-4997	Linux Kernel 3.13	7.8	0.04%	Privilege escalation & Denial of service
CVE-2016-8655	Linux Kernel 3.13	7.8	0.04%	Privilege escalation & Denial of service
CVE-2021-3156	Sudo 1.8.9p5	7.8	96.79%	Code execution & Privilege escalation

CVE-2023-22809	Sudo 1.8.9p5	7.8	0.05%	Privilege escalation
CVE-2019-14287	Sudo 1.8.9p5	9	1.94%	Privilege escalation
CVE-2016-7076	Sudo 1.8.9p5	7.8	0.04%	Code execution
CVE-2019-18634	Sudo 1.8.9p5	7.8	0.27%	Code execution
CVE-2017-3167	Apache 2.4.7	9.8	1.00%	Privilege escalation
CVE-2017-7679	Apache 2.4.7	9.8	0.48%	Code execution
CVE-2018-1312	Apache 2.4.7	9.8	1.92%	Privilege escalation
CVE-2021-26691	Apache 2.4.7	9.8	94.51%	Code execution
CVE-2021-39275	Apache 2.4.7	9.8	3.74%	Code execution
CVE-2021-3492	Ubuntu 14.04.6	8.8	0.05%	Privilege escalation & Denial of service
CVE-2021-3493	Ubuntu 14.04.6	8.8	0.41%	Privilege escalation
CVE-2015-1328	Ubuntu 14.04.6	7.8	0.06%	Privilege escalation

Table 13. Docenti vulnerability report

A fundamental aspect of result evaluation involved risk analysis. This activity allows for a qualitative assessment of the risk associated with each vulnerability to establish priorities for their management within the system. A common approach involves using tables that represent the likelihood of a threat occurring and the potential impact if it were to occur. Specifically, the tables used were the same as in the previous case.

Severity	Impact
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10

Table 14. Docenti impact severity

Severity	Likelihood (%)
Impossible	0.00
Less probable	0.01 - 39.99
Probable	40 - 69.99
High probable	70 - 89.99
Sure	90 - 100

Table 15. Docenti likelihood severity

	None	Low	Medium	High	Critical
Impossible	Inexistent	Inexistent	Inexistent	Inexistent	Inexistent
Less probable	Inexistent	Slight	Slight	Normal	Great
Probable	Inexistent	Slight	Normal	Great	Catastrophic
High probable	Inexistent	Normal	Great	Catastrophic	Catastrophic
Sure	Inexistent	Great	Catastrophic	Catastrophic	Catastrophic

Table 16. Docenti risk matrix

After determining the tables to use for conducting the risk analysis, severity values for impact and probability were assigned to all the examined CVEs.

CVE	Impact	Likelihood
CVE-2016-1908	Critical	Less probable
CVE-2015-5600	High	Less probable
CVE-2016-0778	High	Less probable
CVE-2016-10012	High	Less probable
CVE-2016-10708	High	Less probable
CVE-2016-5195	High	High probable
CVE-2021-22555	High	Less probable
CVE-2015-1328	High	Less probable
CVE-2016-4997	High	Less probable
CVE-2016-8655	High	Less probable
CVE-2021-3156	High	Sure
CVE-2023-22809	High	Less probable
CVE-2019-14287	Critical	Less probable
CVE-2016-7076	High	Less probable
CVE-2019-18634	High	Less probable
CVE-2017-3167	Critical	Less probable
CVE-2017-7679	Critical	Less probable
CVE-2018-1312	Critical	Less probable
CVE-2021-26691	Critical	Sure
CVE-2021-39275	Critical	Less probable
CVE-2021-3492	High	Less probable
CVE-2021-3493	High	Less probable
CVE-2015-1328	High	Less probable

Table 17. Docenti vulnerability severities

Finally, the associated risk for each CVE was determined.

CVE	Risk
CVE-2016-1908	Great
CVE-2015-5600	Normal
CVE-2016-0778	Normal
CVE-2016-10012	Normal
CVE-2016-10708	Normal
CVE-2016-5195	Catastrophic
CVE-2021-22555	Normal
CVE-2015-1328	Normal
CVE-2016-4997	Normal
CVE-2016-8655	Normal
CVE-2021-3156	Catastrophic
CVE-2023-22809	Normal
CVE-2019-14287	Great
CVE-2016-7076	Normal
CVE-2019-18634	Normal
CVE-2017-3167	Great
CVE-2017-7679	Great
CVE-2018-1312	Great
CVE-2021-26691	Catastrophic
CVE-2021-39275	Great
CVE-2021-3492	Normal
CVE-2021-3493	Normal
CVE-2015-1328	Normal

Table 18. Docenti vulnerability risks

Twenty-three vulnerabilities were analyzed. Note that the risk analysis reported medium to high-risk values for most of the identified vulnerabilities. Additionally, there are three cases of catastrophic risk related to the Linux Kernel, Sudo, and Apache.

It's worth noting that, as expected, one of these vulnerabilities is the one that allowed for privilege escalation.

The last activity carried out in this phase involves the generation of the RAV, which is a comprehensive and structured document that provides an overall representation of the system's attack surface. To obtain the RAV associated with the Docenti machine, the Excel spreadsheet provided by ISECOM was used. The compilation of the RAV involved following the guidelines provided in the previous chapter.

Specifically, for the Visibility category, the accounts present within the system were counted, and for the Access category, all the open ports identified in the previous scans were used.

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	
Visibility	25
Access	2
Trust	0
Total (Porosity)	27

OPSEC
11,775361

CONTROLS

Class A		Missing
Authentication	2	25
Indemnification	0	27
Resilience	2	25
Subjugation	0	27
Continuity	0	27
Total Class A	4	131

Class B		Missing
Non-Repudiation	2	25
Confidentiality	1	26
Privacy	0	27
Integrity	1	26
Alarm	2	25
Total Class B	6	129

		True Missing
All Controls Total	10	260
Whole Coverage	3,70%	96,30%

True Controls
4,017304

Full Controls
4,017304

True Coverage A
2,96%

True Coverage B
4,44%

Total True Coverage
3,70%

LIMITATIONS		Item Value	Total Value
Vulnerabilities	1429	10,629630	15189,740741
Weaknesses	4	5,851852	23,407407
Concerns	0	5,777778	0,000000
Exposures	0	54,037037	0,000000
Anomalies	0	53,074074	0,000000
Total # Limitations	1433		15213,1481

Limitations
38,219836

Security Δ
-45,98

True Protection
54,02

Actual Security: 56,5142 ravs

Figure 13. Docenti RAV

4.2.5.4. Hardening

Following the identification of weaknesses and vulnerabilities and their analysis in the previous phases, it is crucial to identify and apply the best possible hardening practices.
The first and most important operation to significantly reduce the number of vulnerabilities affecting the system is to perform all possible updates. This ensures the advancement of the kernel, the operating system, and tools to the latest versions in line with the most up-to-date security measures provided by developers.
Specifically, the elements that most urgently require updating are the Linux Kernel, SSH, and Sudo. To do this, it may be sufficient to update Ubuntu and the installed packages.
As in the previous case, the most effective hardening measure to prevent an attacker from restarting the VM in Recovery Mode to enter or modify the root user's password is to set a boot policy that requires the input of the system administrator's password. To do this, an appropriate configuration file can be placed in the /etc/grub.d directory. It is important to note that files in this directory are usually readable by any user of the system. Therefore, it is necessary to modify these permissions so that only the administrator can read the content of the file and still enter the password to be used as a hashed password.
Another hardening policy to apply involves modifying the configuration of the already fairly secure firewall. Once again, UFW was used for this purpose. The changes to be made include increasing the level of detail of automatically generated log files and denying all outgoing packets, allowing only those of necessary services.
To provide the web service offered by the examined machine, it is advisable to consider transitioning to the HTTPS protocol, which, thanks to certificates, ensures greater security.
For hardening the file that allowed obtaining access credentials to the MySQL database, it is crucial to change its permissions so that users other than the administrator cannot read or modify them.
The most effective hardening for the SSH service is its complete deactivation. However, if it is a fundamental resource, it would be advisable to intervene in its configuration file, namely /etc/ssh/sshd_config. Specifically, the following operations are to be carried out:

- Disabling password authentication: This procedure forces the use of keys (public key, private key) to mitigate brute force attempts. To apply the modification, set the "PasswordAuthentication no" parameter. However, if you want to keep password authentication, it would be advisable to prevent empty passwords using the "PermitEmptyPasswords no" parameter.

- Use of additional tools: To mitigate brute force attacks, you can use tools like Fail2Ban. It monitors failed login attempts from an IP address, and if the number of failed attempts exceeds a certain threshold within a certain time interval, that address is banned for a certain period.
- Changing the default or common SSH port: Most exploits target common SSH ports. Changing the port can reduce the potential number of attacks. You can make this change using the "Port ###" parameter.
- Enabling timeout for inactivity: An SSH connection can remain active without any activity. This poses a security risk, so it's a good practice to configure an inactivity timeout interval. To do this, set the "ClientAliveInterval ###" parameter.
- Enabling access only from certain IP addresses using the "ListenAddress" parameter.

Another hardening practice is to prevent brute force attacks, protecting accounts from unauthorized access using PAM Faillock. Faillock is a PAM module that tracks failed user login attempts and manages account blocking once a certain threshold is exceeded.

A final hardening measure to implement for increased security is to enforce the use of strong passwords for system users and those registered in databases. As evidenced by the lack of success in the brute force practices applied to SSH and the failure of cracking attempts within the databases, the passwords currently in use are already complex enough, but there is no control method to enforce this choice. To encourage the use of robust passwords, you can use the validate_password component of MySQL. This tool sets a policy that a user must follow when choosing their password.

For system users, you can use the libpam-pwquality library, which allows specifying the minimum elements that each password must contain through a configuration file. Finally, another good practice regarding passwords is to set an expiration time so that they are changed periodically.

4.2.6. Virtual machine: RepositoryGit

4.2.6.1. Vulnerability analysis

As in previous cases, various scans were conducted through the Kali Linux VM to assess the security of the virtual machine named RepositoryGit.

The first activity involved identifying open ports. Once again, the Nmap tool was used to perform both types of scans described earlier. The most significant result from the scans, in addition to the open ports used for HTTP and HTTPS protocols, was the identification of an open port associated with the SSH service, revealing the use of OpenSSH version 7.6p1. This information is particularly relevant because this version of OpenSSH is known to be susceptible to 11 documented security vulnerabilities. The most significant vulnerabilities, based on known exploits and potential damage, include:

- CVE-2020-15778: scp in OpenSSH through 8.3p1 allows command injection in the scp.c toremote function, as demonstrated by backtick characters in the destination argument. NOTE: the vendor reportedly has stated that they intentionally omit validation of "anomalous argument transfers" because that could "stand a great chance of breaking existing workflows".
- CVE-2021-41617: sshd in OpenSSH 6.2 through 8.x before 8.8, when certain non-default configurations are used, allows privilege escalation because supplemental groups are not initialized as expected. Helper programs for AuthorizedKeysCommand and AuthorizedPrincipalsCommand may run with privileges associated with group memberships of the sshd process, if the configuration specifies running the command as a different user.
- CVE-2019-6109: An issue was discovered through OpenSSH 7.9. Due to missing character encoding in the progress display, a malicious server (or Man-in-The-Middle attacker) can employ crafted object names to manipulate the client output, e.g., by using ANSI control codes to hide additional files being transferred. This affects refresh_progress_meter() in progressmeter.c.
- CVE-2019-6110: Through OpenSSH 7.9, due to accepting and displaying arbitrary stderr output from the server, a malicious server (or Man-in-The-Middle attacker) can manipulate the client output, for example to use ANSI control codes to hide additional files being transferred.
- CVE-2020-14145: The client side in OpenSSH 5.7 through 8.4 has an Observable Discrepancy leading to an information leak in the algorithm negotiation. This

allows man-in-the-middle attackers to target initial connection attempts (where no host key for the server has been cached by the client). NOTE: some reports state that 8.5 and 8.6 are also affected.

These vulnerabilities, once exploits are discovered to utilize them, can have severe consequences for system security, including privilege escalation and the risk of unauthorized command execution.

Following the identification of open ports related to the HTTP and HTTPS protocols, entering the IP address into the web browser allowed access to the site provided by the machine in question. Once again, the /etc/hosts file was used to map the IP address to the associated domain name. The domain name was obtained through the inspection of the certificate present on the accessed web page. Furthermore, the /etc/environment file was used to create a mapping between the keyword to be used as an environment variable and the domain name.

The identified open ports represent a critical component for system administration and management. Therefore, it is essential to perform targeted analyses to detect potential weaknesses or exposures.

As in the previous case, the Dirb and FFuF tools were used to identify hidden or unauthorized files and directories. However, no significant results were obtained. Due to an automatic redirection policy, it is not possible to discover additional pages beyond those available, which do not provide relevant information.

In order to detect the possible presence of WordPress installations, the WPScan tool was employed. However, as expected, no positive results were obtained.

The last tool used in the security analysis was Nikto, used for identifying potential vulnerabilities and configuration issues within web applications. However, once again, no significant results were obtained.

4.2.6.2. Exploitation

Following the vulnerability analysis phase, attempts were made to find a way to access the machine.

As in previous cases, various avenues were pursued with the aim of gaining access to the system, either by obtaining credentials or by creating a reverse shell impersonating any user.

Initially, brute force attacks were executed against the SSH service and the login page of the website provided by the machine. In both cases, the Hydra tool was used. However, no positive results were obtained, indicating that the passwords used did not match those in common password lists.

Extensive research was conducted to identify possible exploits related to the vulnerabilities identified in the previous phase. However, despite multiple attempts, no exploits were found, and access was not achieved.

Considering a scenario where new exploits were developed for use in the current context, it was decided to simulate what could happen if an attacker managed to gain access as a non-privileged user.

To do this, since it was not possible to access the inside of the machine, revealing a reasonable level of security, an action was taken that leveraged the administrator privileges within the Proxmox virtualization environment. Specifically, a forced reboot was performed in Recovery Mode for the VM under examination. This allowed for the setting of an arbitrary password for the root user to ensure access at any time and the creation of a new non-privileged user.

As in previous cases, once inside the machine, it was possible to navigate through the folders to locate potentially exploitable information. To facilitate this process, the LinPEAS tool was used, which allows for a comprehensive system scan, collecting a vast amount of information about the configuration, running processes, services, system files, and permissions.

Specifically, users with administrator privileges were identified to execute targeted brute force techniques, but they were not successful.

Subsequently, a list of possible executable files with administrator privileges, i.e., those belonging to an administrator with the SUID bit assigned, was provided. However, no significant results were obtained.

Other relevant information provided by the tool included the identification of the Linux Kernel version (4.15), Ubuntu operating system version (18.04.6), and Sudo utility version

(1.8.21p2). To complete the identification of the versions of the relevant system components, specific commands were executed to retrieve the PostgreSQL database version (10.23).

For the Linux Kernel version, 830 vulnerabilities were identified. The most significant vulnerabilities, based on known uses and potential damage, include:

- CVE-2018-20961: In the Linux kernel before 4.16.4, a double free vulnerability in the f_midi_set_alt function of drivers/usb/gadget/function/f_midi.c in the f_midi driver may allow attackers to cause a denial of service or possibly have unspecified other impact.
- CVE-2019-14896: A heap-based buffer overflow vulnerability was found in the Linux kernel, version kernel-2.6.32, in Marvell WiFi chip driver. A remote attacker could cause a denial of service (system crash) or, possibly execute arbitrary code, when the lbs_ibss_join_existing function is called after a STA connects to an AP.
- CVE-2019-14091: A heap overflow flaw was found in the Linux kernel, all versions 3.x.x and 4.x.x before 4.18.0, in Marvell WiFi chip driver. The vulnerability allows a remote attacker to cause a system crash, resulting in a denial of service, or execute arbitrary code. The highest threat with this vulnerability is with the availability of the system. If code execution occurs, the code will run with the permissions of root. This will affect both confidentiality and integrity of files on the system.
- CVE-2019-15292: An issue was discovered in the Linux kernel before 5.0.9. There is a use-after-free in atalk_proc_exit, related to net/appletalk/atalk_proc.c, net/appletalk/ddp.c, and net/appletalk/sysctl_net_atalk.c.
- CVE-2019-14895: A heap-based buffer overflow was discovered in the Linux kernel, all versions 3.x.x and 4.x.x before 4.18.0, in Marvell WiFi chip driver. The flaw could occur when the station attempts a connection negotiation during the handling of the remote devices country settings. This could allow the remote device to cause a denial of service (system crash) or possibly execute arbitrary code.

It is important to note that none of these vulnerabilities have usable exploits.

For the Sudo version, the following vulnerabilities were identified:

- CVE-2019-14287: In Sudo before 1.8.28, an attacker with access to a Runas ALL sudoer account can bypass certain policy blacklists and session PAM modules, and can cause incorrect logging, by invoking sudo with a crafted user ID. For example,

this allows bypass of !root configuration, and USER= logging, for a "sudo -u \#$((0xffffffff))" command.

- CVE-2019-18634: In Sudo before 1.8.26, if pwfeedback is enabled in /etc/sudoers, users can trigger a stack-based buffer overflow in the privileged sudo process. (pwfeedback is a default setting in Linux Mint and elementary OS; however, it is NOT the default for upstream and many other packages, and would exist only if enabled by an administrator.) The attacker needs to deliver a long string to the stdin of getln() in tgetpass.c.
- CVE-2021-3156: Sudo before 1.9.5p2 contains an off-by-one error that can result in a heap-based buffer overflow, which allows privilege escalation to root via "sudoedit -s" and a command-line argument that ends with a single backslash character.
- CVE-2021-23240: selinux_edit_copy_tfiles in sudoedit in Sudo before 1.9.5 allows a local unprivileged user to gain file ownership and escalate privileges by replacing a temporary file with a symlink to an arbitrary file target. This affects SELinux RBAC support in permissive mode. Machines without SELinux are not vulnerable.
- CVE-2023-22809: In Sudo before 1.9.12p2, the sudoedit (aka -e) feature mishandles extra arguments passed in the user-provided environment variables (SUDO_EDITOR, VISUAL, and EDITOR), allowing a local attacker to append arbitrary entries to the list of files to process. This can lead to privilege escalation. Affected versions are 1.8.0 through 1.9.12.p1. The problem exists because a user-specified editor may contain a "--" argument that defeats a protection mechanism.
- CVE-2022-43995: Sudo 1.8.0 through 1.9.12, with the crypt() password backend, contains a plugins/sudoers/auth/passwd.c array-out-of-bounds error that can result in a heap-based buffer over-read. This can be triggered by arbitrary local users with access to Sudo by entering a password of seven characters or fewer. The impact could vary depending on the system libraries, compiler, and processor architecture.
- CVE-2019-18684: Sudo through 1.8.29 allows local users to escalate to root if they have write access to file descriptor 3 of the sudo process. This occurs because of a race condition between determining a uid, and the setresuid and openat system calls. The attacker can write "ALL ALL=(ALL) NOPASSWD:ALL" to /proc/#####/fd/3 at a time when Sudo is prompting for a password.

- CVE-2023-28486: Sudo before 1.9.13 does not escape control characters in log messages.
- CVE-2023-28487: Sudo before 1.9.13 does not escape control characters in sudoreplay output.
- CVE-2021-23239: The sudoedit personality of Sudo before 1.9.5 may allow a local unprivileged user to perform arbitrary directory-existence tests by winning a sudo_edit.c race condition in replacing a user-controlled directory by a symlink to an arbitrary path.

It is important to note that CVE-2023-22809 and CVE-2021-3156 have usable exploits, but they did not produce any results in the current case. Specifically, CVE-2021-3156 due to the absence of the required "make" tool.

Finally, for the Ubuntu and PostgreSQL versions, no vulnerabilities were identified.

To complete this phase and achieve a more comprehensive understanding of the weaknesses, configuration files present in the system were analyzed after gaining access as an administrator. In particular, this analysis revealed suboptimal configurations regarding SSH and VM boot policies. The latter allowed the machine to be restarted in Recovery Mode.

Finally, the last weakness identified was the lack of an active firewall.

4.2.6.3. Outcomes evaluation

Within this phase, a summary and analysis of everything obtained in the previous phases have been carried out. This allows for a comprehensive overview of the situation within the system with greater accessibility to the identified weaknesses and vulnerabilities.

Specifically, the main weaknesses pertain to the following incorrect configurations:

- Machine boot: There are no obstacles to rebooting the machine, allowing it to start in Recovery Mode.
- Firewall: There is no active firewall.
- Web service login: Brute force attacks were possible.
- SSH: Examining the SSH configuration file, insecure policies were identified. Additionally, brute force attacks were possible.

As for the vulnerabilities, just like in the previous case, they have been grouped into a table to display relevant information for each of them.

CVE	Vulnerability source	Potential impact (CVSS)	Exploitation likelihood (EPSS)	Expected consequence
CVE-2020-15778	OpenSSH 7.6p1	7.8	0.29%	Code execution
CVE-2021-41617	OpenSSH 7.6p1	7.0	0.06%	Privilege escalation
CVE-2019-6109	OpenSSH 7.6p1	6.8	0.15%	Code execution
CVE-2019-6110	OpenSSH 7.6p1	6.8	0.34%	Code execution
CVE-2020-14145	OpenSSH 7.6p1	5.9	0.21%	Code execution
CVE-2018-20961	Linux Kernel 4.15	10	2.02%	Code execution & Denial of service
CVE-2019-14896	Linux Kernel 4.15	10	1.38%	Code execution & Denial of service
CVE-2019-14091	Linux Kernel 4.15	10	2.23%	Code execution & Denial of service
CVE-2019-15292	Linux Kernel 4.15	10	0.06%	Code execution
CVE-2019-14895	Linux Kernel 4.15	9.8	0.81%	Code execution & Denial of service
CVE-2019-14287	Sudo 1.8.21p2	9	1.94%	Code execution
CVE-2019-18634	Sudo 1.8.21p2	7.8	0.27%	Code execution
CVE-2021-3156	Sudo 1.8.21p2	7.8	96.79%	Code execution & Privilege escalation

CVE-2021-23240	Sudo 1.8.21p2	7.8	0.08%	Code execution
CVE-2023-22809	Sudo 1.8.21p2	7.8	0.05%	Privilege escalation
CVE-2022-43995	Sudo 1.8.21p2	7.1	0.04%	Code execution
CVE-2019-18684	Sudo 1.8.21p2	7.0	0.04%	Code execution
CVE-2023-28486	Sudo 1.8.21p2	5.3	0.05%	Code execution
CVE-2023-28487	Sudo 1.8.21p2	5.3	0.05%	Code execution
CVE-2021-23239	Sudo 1.8.21p2	2.5	0.05%	Code execution

Table 19. RepositoryGit vulnerability report

A fundamental aspect for evaluating the results has been the risk analysis. This activity allows for a qualitative assessment of the risk associated with each vulnerability to establish priorities in their management within the system. A common approach involves using tables that represent the probability of a threat occurring and the potential impact if it were to occur.

Specifically, the tables used were the same as in the previous case.

Severity	**Impact**
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10

Table 20. RepositoryGit impact severity

Severity	Likelihood (%)
Impossible	0.00
Less probable	0.01 - 39.99
Probable	40 - 69.99
High probable	70 - 89.99
Sure	90 - 100

Table 21. RepositoryGit likelihood severity

	None	Low	Medium	High	Critical
Impossible	Inexistent	Inexistent	Inexistent	Inexistent	Inexistent
Less probable	Inexistent	Slight	Slight	Normal	Great
Probable	Inexistent	Slight	Normal	Great	Catastrophic
High probable	Inexistent	Normal	Great	Catastrophic	Catastrophic
Sure	Inexistent	Great	Catastrophic	Catastrophic	Catastrophic

Table 22. RepositoryGit risk matrix

After determining the tables to use for conducting the risk analysis, severity values for impact and probability were assigned to all the examined CVEs.

CVE	Impact	Likelihood
CVE-2020-15778	High	Less probable
CVE-2021-41617	High	Less probable
CVE-2019-6109	Medium	Less probable
CVE-2019-6110	Medium	Less probable
CVE-2020-14145	Medium	Less probable
CVE-2018-20961	Critical	Less probable
CVE-2019-14896	Critical	Less probable
CVE-2019-14091	Critical	Less probable

CVE-2019-15292	Critical	Less probable
CVE-2019-14895	Critical	Less probable
CVE-2019-14287	Critical	Less probable
CVE-2019-18634	High	Less probable
CVE-2021-3156	High	Sure
CVE-2021-23240	High	Less probable
CVE-2023-22809	High	Less probable
CVE-2022-43995	High	Less probable
CVE-2019-18684	High	Less probable
CVE-2023-28486	Medium	Less probable
CVE-2023-28487	Medium	Less probable
CVE-2021-23239	Low	Less probable

Table 23. RepositoryGit vulnerability severities

Finally, the associated risk for each CVE was determined.

CVE	Risk
CVE-2020-15778	Normal
CVE-2021-41617	Normal
CVE-2019-6109	Slight
CVE-2019-6110	Slight
CVE-2020-14145	Slight
CVE-2018-20961	Great
CVE-2019-14896	Great
CVE-2019-14091	Great
CVE-2019-15292	Great
CVE-2019-14895	Great
CVE-2019-14287	Great
CVE-2019-18634	Normal
CVE-2021-3156	Catastrophic
CVE-2021-23240	Normal
CVE-2023-22809	Normal
CVE-2022-43995	Normal
CVE-2019-18684	Normal
CVE-2023-28486	Slight
CVE-2023-28487	Slight
CVE-2021-23239	Great

Table 24. RepositoryGit vulnerability risks

Twenty vulnerabilities were analyzed. Note that the risk analysis reported medium to high-risk values for most of the identified vulnerabilities. Additionally, there is one case of catastrophic risk related to Sudo.

The final task performed within this phase involves the generation of the RAV, which is a comprehensive and structured document that provides an overall representation of the

system's attack surface. To obtain the RAV associated with the RepositoryGit machine, the Excel spreadsheet provided by ISECOM was used.

The compilation of this document involved following the guidelines provided in the previous chapter. Specifically, for the Visibility category, the accounts present within the system were counted, and for the Access category, all open ports identified during the previous scans were utilized. Additionally, due to the presence of the previously mentioned automatic redirection policy, the Subjugation category was appropriately compiled.

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

OPSEC	
Visibility	4
Access	5
Trust	0
Total (Porosity)	9

CONTROLS		
Class A		Missing
Authentication	2	7
Indemnification	0	9
Resilience	5	4
Subjugation	1	8
Continuity	0	9
Total Class A	8	37

Class B		Missing
Non-Repudiation	5	4
Confidentiality	3	6
Privacy	0	9
Integrity	3	6
Alarm	5	4
Total Class B	16	29

		True Missing
All Controls Total	24	66
Whole Coverage	26,67%	73,33%

LIMITATIONS		Item Value	Total Value
Vulnerabilities	851	8,333333	7091,666667
Weaknesses	4	5,111111	20,444444
Concerns	0	4,222222	0,000000
Exposures	0	95,733333	0,000000
Anomalies	0	95,000000	0,000000
Total # Limitations	855		7112,1111

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	8,730399
True Controls	5,674005
Full Controls	5,674005
True Coverage A	17,78%
True Coverage B	35,56%
Total True Coverage	26,67%
Limitations	34,245894
Security Δ	-37,30
True Protection	62,70

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM
www.osstmm.org

Actual Security: 63,249 ravs

Figure 14. RepositoryGit RAV

4.2.6.4. Hardening

Following the identification of weaknesses and vulnerabilities and their analysis in the previous phases, it is crucial to identify and apply the best possible hardening practices.

The first operation to perform, and arguably the most important in significantly reducing the number of vulnerabilities affecting the system, is to carry out all possible updates. This ensures the advancement of the Linux Kernel, the operating system, and the tools to their latest versions, in line with the most up-to-date security measures provided by developers.

Specifically, the elements that require the most updating are the Linux Kernel, SSH, and Sudo. To do this, updating Ubuntu and the installed packages may suffice.

As in the previous case, the most effective hardening measure to prevent an attacker from rebooting the VM in Recovery Mode to insert or modify the root user's password is to set a boot policy that requires the administrator's password to be entered. To achieve this, an appropriate configuration file can be placed within the /etc/grub.d directory. It's important to note that files within this directory are typically readable by any system user, so it's necessary to modify these permissions so that only the administrator can read the file and enter the password as a hashed password.

Another hardening policy to implement is the activation of the firewall with a secure configuration. In this case, UFW has also been used. After activation, it's essential to set the level of detail for automatically generated log files to medium-high and deny all incoming and outgoing packets, allowing only those from necessary services. Additionally, consider closing the port for the HTTP protocol, leaving only the HTTPS protocol open, which provides greater security through certificates.

The most effective hardening measure for the SSH service is to disable it entirely. However, if it represents a critical resource, it would be advisable to intervene in its configuration file, i.e., /etc/ssh/sshd_config.

Specifically, the operations to be performed are as follows:

- Disable password authentication: This procedure enforces the use of keys (public key, private key) to mitigate brute force attempts. To apply the modification, set the "PasswordAuthentication no" parameter. However, if you wish to retain password authentication, it's advisable to prevent empty passwords using the "PermitEmptyPasswords no" parameter.
- Use additional tools: To mitigate brute force attacks, tools like Fail2Ban can be used. Fail2Ban monitors failed login attempts from an IP address, and if the

number of failed attempts exceeds a certain threshold within a specified time interval, the IP address is banned for a certain period.

- Change the default or common SSH port: Most exploits target common SSH ports. Changing the port can reduce the potential number of attacks. This change can be applied using the "Port ###" parameter.
- Enable inactivity timeout: An SSH connection can remain active without any activity, posing a security risk. Therefore, it's good practice to configure an inactivity timeout interval. This can be done by setting the "ClientAliveInterval ###" parameter.
- Enable access only from specific IP addresses using the "ListenAddress" parameter.

Another hardening practice is to prevent brute force attacks and protect accounts from unauthorized access by using PAM Faillock. Faillock is a PAM module that tracks failed user login attempts and manages account lockout after a certain threshold is exceeded.

A final hardening measure to enhance security is to enforce the use of strong passwords for system users. As evidenced by the lack of success in brute force attempts on SSH and the web interface login page of the VM, the passwords currently in use are already quite complex. However, there is no control method in place to enforce this choice. To encourage the use of strong passwords, the libpam-pwquality library can be used. This tool allows specifying the minimum requirements for each password through a configuration file. Lastly, another good practice regarding passwords is to set an expiration time so that passwords are periodically changed.

4.3. Evaluation of obtained results and future improvements

Following the application of the designed methodology to the examined systems, multiple significant results were identified. In particular, a sufficient level of security against external threats was highlighted. Indeed, in all systems, it was not possible to gain access through the exploitation of exploits or brute force techniques. However, the absence of active firewalls, security-oriented configurations, and the presence of outdated elements represent significant weaknesses even over time.

From the perspective of internal threats, significant shortcomings were identified. In fact, in two of the four analyzed systems, namely Robotcup and Docenti, it was possible to access sensitive information. Furthermore, in the second case, it was also possible to exploit an exploit that allowed for privilege escalation. This indicates an immediate need for intervention.

To validate the effectiveness of the proposed hardening measures, following their implementation, the entire methodology was applied again to the examined systems. This allowed for the calculation of new RAVs to compare them with the previous ones to check for differences in attack surfaces. Specifically, the most restrictive hardening measures possible were adopted, including the closure of unnecessary ports (i.e., those related to SSH and HTTP).

The new RAV associated with Proxmox virtual environment is as follows:

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC

Visibility	2
Access	1
Trust	0
Total (Porosity)	3

CONTROLS

Class A		Missing
Authentication	2	1
Indemnification	0	3
Resilience	1	2
Subjugation	0	3
Continuity	0	3
Total Class A	3	12

Class B		Missing
Non-Repudiation	1	2
Confidentiality	1	2
Privacy	0	3
Integrity	1	2
Alarm	1	2
Total Class B	4	11

		True Missing
All Controls Total	7	23
Whole Coverage	23,33%	76,67%

LIMITATIONS		Item Value	Total Value
Vulnerabilities	118	8,666667	1022,666667
Weaknesses	0	5,000000	0,000000
Concerns	0	4,666667	0,000000
Exposures	0	40,100000	0,000000
Anomalies	0	39,333333	0,000000
Total # Limitations	118		1022,6667

OPSEC	6,143292
True Controls	3,427157
Full Controls	3,427157
True Coverage A	20,00%
True Coverage B	26,67%
Total True Coverage	23,33%
Limitations	25,097478
Security Δ	-27,81
True Protection	72,19

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM
www.osstmm.org

Actual Security: 72,6575 ravs

Figure 15. Proxmox hardened RAV

The new RAV associated with Robotcup virtual machine is as follows:

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	
Visibility	4
Access	1
Trust	0
Total (Porosity)	5

CONTROLS		
Class A		Missing
Authentication	2	3
Indemnification	0	5
Resilience	1	4
Subjugation	0	5
Continuity	0	5
Total Class A	3	22

Class B		Missing
Non-Repudiation	1	4
Confidentiality	1	4
Privacy	0	5
Integrity	1	4
Alarm	1	4
Total Class B	4	21

		True Missing
All Controls Total	7	43
Whole Coverage	14,00%	86,00%

LIMITATIONS		Item Value	Total Value
Vulnerabilities	118	9,600000	1132,800000
Weaknesses	0	5,400000	0,000000
Concerns	0	5,200000	0,000000
Exposures	0	24,460000	0,000000
Anomalies	0	23,600000	0,000000
Total # Limitations	118		1132,8000

OPSEC	7,289124
True Controls	3,427157
Full Controls	3,427157
True Coverage A	12,00%
True Coverage B	16,00%
Total True Coverage	14,00%
Limitations	25,544504
Security Δ	-29,41
True Protection	70,59

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM
www.osstmm.org

Actual Security: 71,3302 ravs

Figure 16. Robotcup hardened RAV

The new RAV associated with Docenti virtual machine is as follows:

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC

Visibility	25
Access	1
Trust	0
Total (Porosity)	26

CONTROLS

Class A		Missing
Authentication	2	24
Indemnification	0	26
Resilience	1	25
Subjugation	0	26
Continuity	0	26
Total Class A	3	127

Class B		Missing
Non-Repudiation	1	25
Confidentiality	1	25
Privacy	0	26
Integrity	1	25
Alarm	1	25
Total Class B	4	126

		True Missing
All Controls Total	7	253
Whole Coverage	2,69%	97,31%

LIMITATIONS		Item Value	Total Value
Vulnerabilities	118	10,730769	1266,230769
Weaknesses	0	5,884615	0,000000
Concerns	0	5,846154	0,000000
Exposures	0	5,511538	0,000000
Anomalies	0	4,538462	0,000000
Total # Limitations	118		1266,2308

OPSEC	11,663184
True Controls	3,427157
Full Controls	3,427157
True Coverage A	2,31%
True Coverage B	3,08%
Total True Coverage	2,69%
Limitations	26,035673
Security Δ	-34,27
True Protection	65,73

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM
www.osstmm.org

Actual Security: 67,4729 ravs

Figure 17. Docenti hardened RAV

The new RAV associated with RepositoryGit virtual machine is as follows:

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	
Visibility	4
Access	1
Trust	0
Total (Porosity)	5

OPSEC 7,289124

CONTROLS		
Class A		Missing
Authentication	2	3
Indemnification	0	5
Resilience	1	4
Subjugation	1	4
Continuity	0	5
Total Class A	4	21

True Controls 3,642315

Full Controls 3,642315

True Coverage A 16,00%

Class B		Missing
Non-Repudiation	1	4
Confidentiality	1	4
Privacy	0	5
Integrity	1	4
Alarm	1	4
Total Class B	4	21

True Coverage B 16,00%

Total True Coverage 16,00%

		True Missing
All Controls Total	8	42
Whole Coverage	16,00%	84,00%

OSSTMM Open Source Security Testing Methodology Manual
OSSTMM www.osstmm.org

LIMITATIONS		Item Value	Total Value
Vulnerabilities	118	9,400000	1109,200000
Weaknesses	0	5,200000	0,000000
Concerns	0	5,200000	0,000000
Exposures	0	24,440000	0,000000
Anomalies	0	23,600000	0,000000
Total # Limitations	118		1109,2000

Limitations 25,452164

Security Δ -29,10

True Protection 70,90

Actual Security: 71,5637 ravs

Figure 18. RepositoryGit hardened RAV

As can be observed, each analyzed system has shown significant improvement. In particular, the most at-risk machine, namely Docenti, achieved an approximately 20% increase in the RAV, indicating a noticeable reduction in the attack surface.

As described earlier, the ideal value for an RAV is 100, but in real-world contexts, reaching this value is not possible. Indeed, as you can see when comparing the Docenti machine to the others, the higher the number of accounts within a system, i.e., the value associated with the Visibility category, and the greater the number of open ports exposing a service, i.e., the value associated with the Access category, the lower the RAV value will be.

Furthermore, it's worth noting that for all the analyzed systems, the update of the operating system led to the use of Linux Kernel 6.2, which is not the latest version available and is subject to 118 known vulnerabilities. In fact, upgrading to the latest version, namely 6.5, is discouraged due to the lack of guarantees and support in case of malfunctions.

The practice of keeping a system up-to-date for security purposes remains valid, but a trade-off between updates and operational stability must be achieved.

The results obtained have validated the methodology designed for performing security assessments and subsequent hardening of computer systems. Indeed, it allowed for the detection and provision of the best possible corrections for vulnerabilities and threats compromising the security of these systems.

In particular, the design choices made during the thesis activity have proven to be consistent with the set objectives. It has been verified that the design contributions, which consist of conferring the process with the characteristics of good structure, generalizability, iterativity, replicability, and coherence, perfectly satisfy the needs of Cybersecurity, including flexibility and adaptability to the context.

In particular, the structure divided into well-defined phases was crucial in clearly and precisely defining the operations to be carried out to successfully apply the methodology.

Generalizability allowed the process to be applied to different systems without the need for modifications.

Iterativity was fundamental for validating the effectiveness of the implemented countermeasures, allowing for the comparison of results obtained with previous ones. Moreover, this characteristic enables the security policies to be constantly updated.

Finally, replicability and coherence enable the methodology to be replicated on the same system and in the same context, ensuring the same results are obtained.

A crucial role was played by the guidelines provided by OSSTMM, acquired through an in-depth study of the latter.

An important future improvement involves the development of a tool capable of automating a significant portion of the designed methodology. Specifically, this tool should automatically execute the vulnerability analysis, exploitation, and outcomes evaluation phases, ensuring the generation of a detailed report of all identified vulnerabilities and weaknesses. Finally, for each security threat, the recommended countermeasure should be provided to facilitate the hardening phase.

Bibliography

[1] F. Bazargan, C. Y. Yeun, M. J. Zemerly, "State-of-the-Art of Virtualization, its Security Threats and Deployment Models", in International Journal for Information Security Research (IJISR), 2012

[2] D. M. Zoughbi, N. Dutta, "Hypervisor Vulnerabilities and Some Defense Mechanisms, in Cloud Computing Environment", in International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2020

[3] Institute for Security and Open Methodologies (ISECOM), "Open Source Security Testing Methodology Manual 3", 2010

Webliography

https://pve.proxmox.com/pve-docs/pve-admin-guide.html

https://www.kali.org/tools/

https://nmap.org/book/man.html

https://wpscan.com/wordpresses

https://nvd.nist.gov/vuln-metrics/cvss

https://www.first.org/epss/model

https://www.cvedetails.com/index.php

Acknowledgements

Gratefulness goes to Prof. Vincenzo Carletti, my First Supervisor, who has continuously supported my work, offering invaluable theoretical and practical insights. His guidance has been a beacon, illuminating the path of our research endeavors.

Profound appreciation is extended to Prof. Pasquale Foggia, my Second Supervisor, who effectively motivated and encouraged me throughout the project's realization. His expertise and dedication have been instrumental in shaping the direction of this research.

To my esteemed colleague and fellow traveler, Allegra Cuzzocrea, I extend my deepest gratitude for the exceptional collaboration, unwavering support, boundless availability, and enduring faith in this transformative journey. Her unwavering enthusiasm and tireless efforts have significantly enriched and elevated this academic experience, imbuing it with a vibrant tapestry of ideas and shared dedication. I value her insightful contributions and steadfast encouragement throughout this research endeavor, which have undoubtedly shaped the outcome of this thesis.

I would like to express my heartfelt gratitude to all my colleagues and friends who provided unwavering support throughout my study course. Your encouragement, insightful discussions, and shared experiences have been invaluable in shaping my academic journey and the completion of this thesis. I am truly fortunate to have such an amazing network of individuals who have been a source of inspiration and motivation. Thanks for being a vital part of this significant chapter in my life.

In addition, I express my gratitude to my family, the cornerstone of my journey. My parents, my siblings, my grandmother, and my aunts and uncles have been a source of unwavering support and belief in my abilities. Their encouragement propelled me forward during challenging times and fueled my determination to succeed. Their love and belief in me have been my greatest strength.

Last but certainly not least, I would like to thank all the professors and mentors who have contributed to my academic and personal growth. Your wisdom, encouragement, and constructive criticism have been instrumental in shaping my perspective and honing my skills. This academic achievement would not have been possible without your guidance and belief in my potential.