### **1. Introduction**

The dynamic nature of cybersecurity necessitates ongoing vigilance and proactive strategies to protect digital systems from emerging threats and vulnerabilities. **Penetration testing**, also known as **ethical hacking**, plays a crucial role in this effort by providing organizations with valuable insights into the strength and defenses of their security measures. By simulating real-world cyber-attacks, penetration testing evaluates the resilience of a system’s defenses, identifying potential weaknesses before malicious actors can exploit them.

### **1.1 Objective and Scope**

There will be a network topology with different machines in it. This report details apenetrationtest conducted on the target system on the network **pWnOS**, with the primary goal of assessing its security posture. The test aims to:

* Identify security weaknesses through simulated attacks.
* Exploit vulnerabilities to demonstrate potential risks.
* Provide actionable remediation strategies.

The scenario assumes an **internal user compromise**, mimicking how attackers might infiltrate a system. The ultimate objective is to gain **root access** and locate a file named **proof.txt** (if present) in the root directory, serving as evidence of successful exploitation.

### **2. Summary and Recommendations**

#### **2.1 Summary**

The penetration testing process revealed several critical vulnerabilities within the **pWnOS** environment, exposing potential security risks. These issues spanned from weak authentication mechanisms to out-of-date software, both of which create loopholes for unauthorized privilege and system threat.

One predominant and concerning finding was the **lack of strong user access controls and weak authentication/password policies**, which posed a significant security risk. The **Webmin interface**, in particular, was not adequately protected by strong authentication methods or **Multi-Factor Authentication (MFA)**, making it vulnerable to unauthorized access. This highlights the need for **robust privilege/access control measures**, including MFA and strict password management practices, to enhance security.

Additionally, the presence of **exploitable vulnerabilities** in outdated system services was another major concern. These loopholes could be leveraged by attackers to gain access to sensitive data or compromise the system's integrity.

Another critical issue was the **absence of persistent security patches and updates**, which significantly weakened **pWnOS's** security posture. Failing to promptly deal with known vulnerabilities in a timely manner increases the risk of systems being successfully compromised by attackers, making the system more susceptible to threats.

#### **2.2 Recommendations**

To enhance the security of **pWnOS**, the following measures should be implemented:

* **Consistent Security and Vulnerability Assessment**: Conduct **frequent security audits** and vulnerability assessments to proactively identify and address weaknesses before they can be exploited.
* **Strengthen Access Controls & Implement MFA**: Enforce strict access control policies and require **Multi-Factor Authentication (MFA)** for sensitive services like **Webmin** to prevent unauthorized access.
* **Enhance Security Awareness Among Users**: Provide **ongoing cybersecurity training** to system users, emphasizing best practices such as locking screens, securing workstations, and turning off computers before leaving work to minimize security risks.
* **Enforce a Strong Password Policy**: Require passwords to be **10–12 characters long**, containing **both uppercase and lowercase letters**, at least **one special-case character, and a digit**. When combined with MFA, this will significantly reduce the likelihood of credential-based attacks.
* **Ensure Regular Updates & Patch Management**: Keep all system services and applications updated to protect against known security threats and vulnerabilities.

### **3. Methodology**

There are several recognized **penetration testing methodologies**, each providing structured guidelines for conducting security assessments. According to **Teaganne Finn of IBM (2024)**, the five most widely used methodologies include:

1. **Open-Source Security Testing Methodology Manual (OSSTMM)**
2. **Information System Security Assessment Framework (OISSG)**
3. **Open Web Application Security Project (OWASP)**
4. **National Institute of Standards and Technology (NIST)**
5. **Penetration Testing Execution Standard (PTES)**

For this penetration test, the **OSSTMM methodology** was selected due to its **scientific approach** and adaptability. This methodology provides well-structured, accessible guidelines for testers, ensuring a thorough and systematic assessment.

The specific approach followed in this assessment includes:

1. **Information Gathering** – Collecting details about the target system to identify potential entry points.
2. **Scanning and Mapping** – Analyzing the network structure and identifying active hosts.
3. **Enumeration** – Extracting information about services, users, and configurations.
4. **Gaining Access** – Exploiting vulnerabilities to penetrate the system.
5. **Privilege Escalation (or Denial of Service)** – Elevating privileges to gain deeper access or simulating a **Denial of Service (DoS) attack** to assess system resilience.

This structured methodology ensures a comprehensive evaluation of **pWnOS's** security weaknesses, helping to identify and mitigate risks effectively.

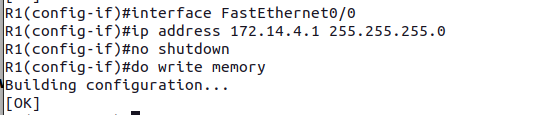
### **4. Network Design and Topology**

The Apporto environment was used to set up the network. Following the specification guide, a switch, a router (c7200), windows machine, kali machine and pWnOS were combined to form a network. The router was configured to act as the DHCP (Dynamic Host Configuration Protocol) server with the IP block 172.16.4.0/24. The following outlines how this was done:

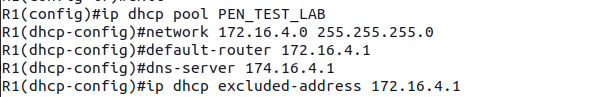
**Configuring the router terminal**



**Assigning the router interface**

ex

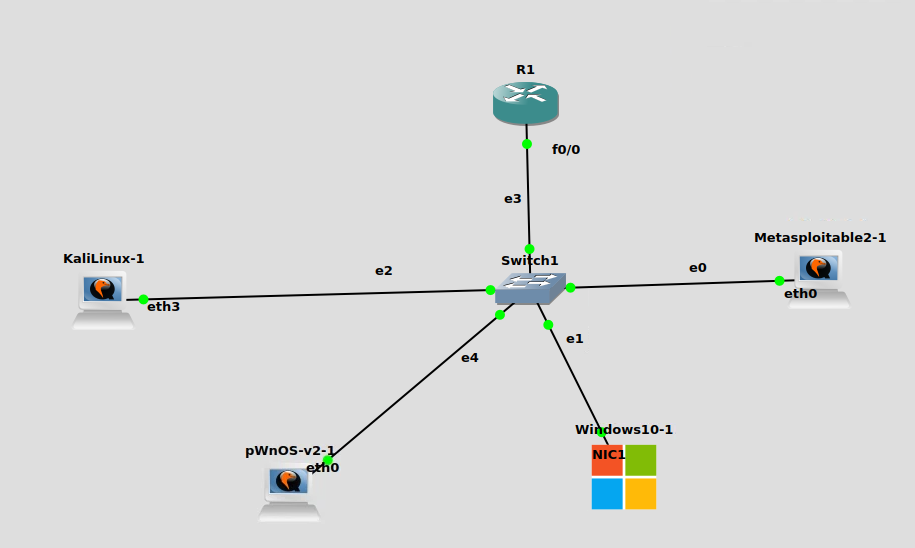
**Configuring the DCHP pool**



**Saving the configuration**



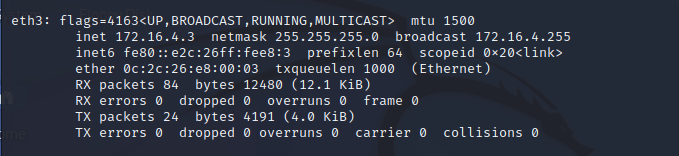
**Final network design and set up**



Switch information:  

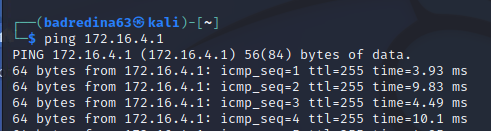

### **5. Information Gathering**

With the network topology done, the tester set up the kali machine to be used as the attaching VM, by creating and configuring the user as **badredina63**. The thing the tester did was verify if the router and network is set up properly and actively running. To do this, the IP of the target machine (kali) was checked with the command **ifconfig**, and it showed **172.16.4.3** as expected because from the network diagram, it was configured on **eth3** to the switch.

Verify on kali  
  


Now, to test if the DHCP server is up, a ping was used which successfully received messages from the router

Ping and test server:

  
  
**all devices and mac addresses**:  
0c:8e:c2:06:00:00 - windows

0c:1a:1a:87:00:00 - pwnos

0c:fc:57:b2:00:00 – metasploitable

With the preliminary verification above confirmed, it is time to discover the machines in the network to find the target. The command below was used to achieve this:

Table 1 (discovering IPs in the network)

|  |
| --- |
| sudo netdiscover -r 172.16.4.0/24 |

The **netdiscover** command requires superuser (**sudo**) permissions to grant privilege to network associated information that normal users are not permitted to. “**netdiscover**” is a CLI (Command Line Utility) that comes with kali to discover all devices on the network. The -r flag details the range of IP addresses configured on the network (172.16.4.0/24). The figure below shows the result after executing the command above:  
   
pic here:

After the tabular list from the result, the reporter identified the target is the machine with MAC address as 3423 and IP address as 234234. To verify if this is reachable, a **ping** was done which confirmed it is up as seen below:

With this discovery, the next step was to collect information about the target machine. The aim is to remain unknown and stealth in collection the information about the device on the network so as to reduce any traces and suspicions on the firewalls. To achieve this, nmap command was used with the flag “-sS”.

Table 2 (being stealth in gathering info)

|  |
| --- |
| sudo nmap -sS 192.168.1.78 |

The command shows opened ports running on the target VM.

Pic here:  
Nonetheless, collection information about opened and running ports is not enough, more information such as ssh-host key information, authentication level and operation system is needed about the target. To achieve this, the -A flag is used in combination with nmap. To additionally achieve non-suspicions and quietness in the network, random 100 IP addresses were spawned which will act as a decoy, making it much harder to reveal the actual IP address collecting the information. The command **-D RND:100** was used to achieve this.   
  
Table 3 (detailed information)

|  |
| --- |
| sudo nmap -A -D RND:50 192.168.1.78 |

The result of this is presented below:

|  |  |
| --- | --- |
| Target VM Details : pWnOS | |
| IP Address |  |
| Domain Name |  |
| Operating System |  |
| Host status |  |
| Authentication Level |  |
| Traceroute |  |
| Opened Ports |  |
| Ssh-host key info: |  |

### **6. Scanning and Mapping**

After information gathering, the target device had its port opened for the following numbers – 445, 80, 22, 10000.

**Port 445 Scan**

The Server Message Block protocol (SMB) typically runs on port 445 (ref). The SMB is primarily used for network services such as printer and file sharing. With this in mind, the tester explored the port to scan for any vulnerability and exploits in SMB. The Metasploit software was used. According to ref, Metasploit is a widely used, open-source penetration testing framework that helps security professionals identify and exploit vulnerabilities in networks, systems, and applications by providing a collection of tools, exploits, payloads, and auxiliary functions.

To identify all SMB exploits/vulnerabilities available in Metasploit, the reporter executed the command:  
  
  
Table 3 (SMB command)

|  |
| --- |
| grep scanner search smb |

From the results listed in below (my kali username can’t be shown so verify using the IP addresses), the **auxiliary/scanner/smb/smb\_ms17\_010** module was chosen. This module is associated with the well-known **EternalBlue** vulnerability (**MS17-010**) in Microsoft's **SMB protocol**, which was leaked in April 2017 by a group called **Shadow Brokers**. This vulnerability enables **remote access and code execution** on susceptible systems.

A screenshot of a computer program

AI-generated content may be incorrect.

Nonetheless, after configuring the necessary options for the module, the tester discovered that the pWnOS VM did not seem vulnerable to SMB exploits, as shown below.

A screenshot of a computer program

AI-generated content may be incorrect.

**Port 22 and 80 scan**

Port 22 is running the Secure Shell protocol (SSH), suggesting that the user may have remotely accessed their organization's system for administrative tasks or network sharing like file, fax or printer transfers. This could be useful for the tester when attempting to gain access to the system.

Additionally, the presence of an open port 80 indicates that an HTTP server is active. Since HTTP is associated with web services, the tester used Kali’s inbuilt Firefox web browser to scan and gather additional information.

The homepage of the web server displayed the message **"Welcome to pWnOS homepage!"**. Clicking "Next" redirected me to the page shown in the figure below.

Landing page pic here:

**"Please Help"** button was unhelpful because after clicking on it, I was redirected to a page mocking the user with the message:  
*"HAHAHA! Badredin, for a nOOb you REALLY SUCK!"*

Nonetheless, after exploring the page further, the tester noticed that the website URL included query parameters for **connect and help**, both of which have their values as **true**. Upon changing these parameters to **false**, the page instead displayed an error, as shown in the figure below.

pic here

This suggested that additional pages, such as **index1.php**, might exist within the website and could be scanned and accessed. To explore these pages, the command shown in Table 5 was executed.

Table 3 (exploring pages)

|  |
| --- |
| dirb http://172.16.4.2/ |

Dirb is a web content enumeration and discovery tool that conducts directory brute-forcing to identify hidden directories and files on a web server (Reff, 2017). The outcome of this process is shown below.

pic here

The tester explored all the pages but did not find any immediate information that could be used to compromise the user's system. However, some findings might become relevant in later stages of the assessment. For instance, the **index.php** directory contained two subdirectories — **phpMyAdmin and ParentDirectory** —both of which needed login details as seen below:  
A screenshot of a computer

AI-generated content may be incorrect.

Returning to the site's landing page on port 80, the tester observed that modifying the parameters of **help** and **connect** resulted in an error message (refer to Figure 6 above). This suggested a potential avenue for further investigation. Additional research, referencing Tim Fisher (2023), indicated that web pages displaying error messages might be susceptible to vulnerabilities such as **Path Traversal, SQL Injection, and File Inclusion**. These potential risks will be examined further in the **enumeration** section of the report.

**Port 10000 scan**

The port, **10000**, is associated with **MiniServ**, the web server used by **Webmin**.

As a brief summary, according to **Jamie Cameron (2023)**, **Webmin** is a web and UI based, opened source control panel designed to simplify system administration tasks on **Unix-like systems** (such as **Linux**), allowing users to manage servers and services through a web browser interface instead of the command line.

Research on Webmin and its vulnerabilities uncovered a security issue documented in **CVE-2006-3392**, affecting Webmin versions prior to **1.290 version**. This vulnerability stems from improper processing of paths in files (both absolute and relative paths) and HTML decoding, where the **simplify\_path** method is invoked before processing the HTML body. As a result, remote attackers can exploit this flaw to access files on the system. By using crafted sequences such as **"..%01"**, attackers can essentially skip the removal of **"../"** directory traversal sequences when followed by special bytes like **"%01"**, leading to unauthorized file access (**NVD, 2024**).

Next, the tester needed to verify if the Webmin version was earlier than 1.290. To accomplish this, I installed **"searchsploit"**, a command-line tool from the Exploit Database (EDB) project, which helps search for and retrieve information about vulnerabilities. The installation was done using: ***sudo apt install exploitdb***

Once installed, the tester executed the following command to check the Webmin version:

***searchsploit webmin***

This command returned a list of available exploits, including the Webmin versions they affect, as shown in the figure below:

pic here  
The yellow highlight in the figure confirms that the Webmin version is below 1.290, making it potentially vulnerable to this exploit. This will be explored further in the section below.

### **7. Enumeration**

At this stage, the testing requires proactively querying systems and services on the target to collect information, including shared resources, user accounts, , file systems, and configurations, following the scanning and mapping process.

During the mapping and scanning phase, it was identified that the **index.php** file might be susceptible to **File Inclusion** vulnerabilities, as error messages were triggered when modifying its URL parameters. File Inclusion vulnerabilities arise when an application permits an attacker to include a file by specifying its path through a URL or form submission (Derda, 2022)

The tested attempted to retrieve the **/etc/passwd** file from a PHP application running on port 80 of host (pWnOS). The **/etc/passwd** file in Linux and other Unix-like systems is a plain text file that stores basic information about user accounts, including usernames, user IDs (UIDs), group IDs (GIDs), home directories, and the default shell (ref).

The tester attempted this by modifying the connect parameter in the URL to /etc/passwd, which successfully returned a list of user accounts present on the system, as shown below.

pic here

With this information in hand, details such as usernames and shell types could later be leveraged in password cracking once the hashes of the passwords are obtained.

Continuing the assessment, since the website was confirmed to be vulnerable to File Inclusion, an attempt was made to access the **/etc/shadow** file. This file contains sensitive authentication-related data, including hashed password and permissions on files.

To accomplish this, the tester revisited Webmin (running on port 10000) and exploited the vulnerability in the specific Webmin version used by the target system. By utilizing the sequence ..%01..%01..%01..%01..%01/**etc/shadow**, as previously discussed in the mapping section, the tester successfully retrieved the following information:

pic here

Now that both the /etc/shadow and /etc/passwd files are accessible, the next step is to attempt to gain access to the host.

### **8. Gaining Access**

This pen-testing phase focuses on exploiting the vulnerabilities identified during the previous stages and using them to obtain access to the target.

The tester utilized John the Ripper, a password cracking tool which comes pre-installed on kali, with the *passwd* and *shadow* files. By employing various cracking techniques such as dictionary, rule-based and brute-force attacks, John the Ripper aims to decipher plain text passwords from the password hashes saved in the /etc/*shadow* file, which is the tester’s objective at this point.

To proceed, the tester saved the contents of the *etc/shadow* and *etc/passwd* files to the kali machine. These files were copied into *shadow.txt* and *password.txt* on the desktop using the nano text editor. Nano is a command-line text editor commonly found in Unix-like operating systems, including Linux and macOS, making it a user-friendly tool for quick text file editing directly from the terminal.

The figure below shows that these are created and contains required contents:

pic here

By default, **John the Ripper (JTR)** cannot process the raw /etc/passwd and /etc/shadow files directly. To make them compatible, the **unshadow** tool was used to merge the two files into a single formatted file named **JTR-input.txt**, which JTR can read and analyse. This step was executed using the command shown below:

Table 7 (unshadow files)

|  |
| --- |
| unshadow password.txt shadow.txt > JTR-input.txt |

Additionally, the command below is what cracks the password by reverting the hashes

Table 8 (cracking password with John)

|  |
| --- |
| John JTR-input.txt --wordlist=/usr/share/wordlists/rockyou.txt |

According to the John the Ripper’s documentation, the tool uses /usr/share/wordlists/rockyou.txt as its default dictionary attack wordlist. When executing the command, John compares each word from this wordlist against the hashes in JTR-input.txt, attempting to find matching password strings.

Since rockyou.txt is stored in a compressed format (rockyou.txt.gz), it must first be decompressed before use. The tester employed the **gunzip** command to extract the wordlist, making it accessible for the password-cracking process. This step ensures the file path becomes valid and John can properly read the wordlist contents during the attack.

pic here

After executing the command listed in the table above, the password-cracking process completed in **4 minutes and 8 seconds**, successfully revealing the credentials:

* **Username:** vmware
* **Password:** h4ckm3

The results are displayed in the screenshot below for verification.

**Minor Key Insights**:

* The cracked password (h4ckm3) indicates poor password hygiene, emphasizing the need for stronger policies (e.g., complexity requirements, MFA).
* Total cracking time (~4 minutes) underscores the risk of using predictable passwords.

pic here

Using the credentials obtained to log into the pWnOS host successfully worked as shown below:

pic here

Now, the tester needed to **remotely access the target host**, as in a real-world scenario, the target would not be a virtual machine in apporto that I can log into.

This is where the **SSH (Secure Shell) protocol** proved essential. As previously mentioned in the scanning and mapping stage, SSH enables secure remote logins, command execution, file transfers, and administrative tasks.

To establish remote access, the tester used the following command:

Table 9 (ssh login)

|  |
| --- |
| ssh HostKeyAlgorithms%3Dssh-rsa,ssh-dss%20vmware@172.16.4.2 |

This command initiates an **SSH connection** to the pWnOS VM at **192.168.1.78** with the username **"vmware"**. It enforces the use of **RSA/DSA** cryptographic algorithms for host key authentication, ensuring secure communication.

**Key Details:**

* **RSA (Rivest-Shamir-Adleman)** and **DSA (Digital Signature Algorithm)** are encryption methods used for SSH authentication.
* **HostKeyAlgorithms** specifies the permitted host key types:
  + ssh-rsa → Prefers RSA keys.
  + ssh-dss → Allows DSA keys as a fallback.
* The SSH client automatically selects the best-supported algorithm based on the server’s capabilities.

After entering the password (**h4ckm3**), the tester successfully gained remote access to the target system, as shown in the output.

**Simplified Breakdown:**

1. **Command Purpose**: Securely log into 192.168.1.78 as user vmware.
2. **Security Enforcement**: Uses only RSA/DSA keys for authentication.
3. **Result**: Access granted after password verification.

pic here

Having secured a basic level of access, the next objective was privilege escalation.