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!
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 !**
## Detection Details

<table>
<thead>
<tr>
<th>DETECTION</th>
<th>DETAILS</th>
<th>BEHAVIOR</th>
<th>COMMUNITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ikerus</td>
<td>Trojan.PowerShell.Rozenz</td>
<td>Undetected</td>
<td>Kaspersky</td>
</tr>
<tr>
<td>Microsoft</td>
<td>TrojanDownloader.PowerShell.Genihv.A</td>
<td>Undetected</td>
<td>Sophos AV</td>
</tr>
<tr>
<td>ZoneAlarm by Check Point</td>
<td>HEUR.Trojan.PowerShell.Generic</td>
<td>Undetected</td>
<td>Ad-Aware</td>
</tr>
<tr>
<td>AegisLab</td>
<td>Undetected</td>
<td>Avira (no cloud)</td>
<td></td>
</tr>
<tr>
<td>ALYac</td>
<td>Undetected</td>
<td>Antiy-AVL</td>
<td></td>
</tr>
<tr>
<td>Arcbit</td>
<td>Undetected</td>
<td>Avast</td>
<td></td>
</tr>
<tr>
<td>AVG</td>
<td>Undetected</td>
<td>Avira (no cloud)</td>
<td></td>
</tr>
<tr>
<td>Baidu</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BitDefenderTheta</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAT-QuickHeal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cyren</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yanda</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>McAfee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zemana AntiMal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZoneAlarm by Check Point</td>
<td>HEUR.Trojan.PowerShell.Generic</td>
<td>Undetected</td>
<td>Ad-Aware</td>
</tr>
</tbody>
</table>

### Payload Basic Script
```powershell
Set-StrictMode -Version 2

$x211 = 0
using System;
using System.Runtime.InteropServices;

namespace qvn {
  public class func {
    [Flags] public enum AllocationType {
      Commit = 0x1000, Reserve = 0
    }
    public enum MemoryProtection {
      Write = 0x40
    }
    public enum lump {
      Free = 0, Reserve = 0
    }
    public static extern IntPtr VirtualAllocEx(IntPtr hWnd, IntPtr lpAddress, uint dwSize, AllocationType dwType, MemoryProtection dwProtection);"
<table>
<thead>
<tr>
<th>DETECTION</th>
<th>DETAILS</th>
<th>BEHAVIOR</th>
<th>COMMUNITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-Aware</td>
<td>Undetected</td>
<td>Aegislab</td>
<td>Undetected</td>
</tr>
<tr>
<td>Ahnlab-V3</td>
<td>Undetected</td>
<td>ALYac</td>
<td>Undetected</td>
</tr>
<tr>
<td>Anti-XAV</td>
<td>Undetected</td>
<td>Arcabit</td>
<td>Undetected</td>
</tr>
<tr>
<td>Avast</td>
<td>Undetected</td>
<td>Avast-Mobile</td>
<td>Undetected</td>
</tr>
<tr>
<td>AVG</td>
<td>Undetected</td>
<td>Baidu</td>
<td>Undetected</td>
</tr>
<tr>
<td>BitDefender</td>
<td>Undetected</td>
<td>BitDefenderTheta</td>
<td>Undetected</td>
</tr>
<tr>
<td>Bkav</td>
<td>Undetected</td>
<td>CAT-Quick Heal</td>
<td>Undetected</td>
</tr>
<tr>
<td>ClamAV</td>
<td>Undetected</td>
<td>CMC</td>
<td>Undetected</td>
</tr>
<tr>
<td>Comodo</td>
<td>Undetected</td>
<td>Cynet</td>
<td>Undetected</td>
</tr>
<tr>
<td>Cyren</td>
<td>Undetected</td>
<td>DrWeb</td>
<td>Undetected</td>
</tr>
<tr>
<td>Emsisoft</td>
<td>Undetected</td>
<td>elScan</td>
<td>Undetected</td>
</tr>
<tr>
<td>ESET-NO32</td>
<td>Undetected</td>
<td>F-Prot</td>
<td>Undetected</td>
</tr>
<tr>
<td>F-Secure</td>
<td>Undetected</td>
<td>FireEye</td>
<td>Undetected</td>
</tr>
</tbody>
</table>

12.70 KB  2020-08-14 10:12:30 UTC  3 minutes ago
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
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://github.com/nccgroup/demiguise)
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
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

### Steps:

1. **PowerShell HTA setup**
   - Launch HTA payload on the target system.

2. **Browser hooking**
   - Use BeEF to hook the browser and inject code.

3. **C2 handler and... Session opening */o/**
   - Establish communication with the C2 server using BeEF.

---

**One Click RCE**

- HTA payload delivery through a single click can lead to remote code execution (RCE).

---

**Details**

- **Exploit (windows/misc/hta_server)**
- **Sessions**

---

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Type</th>
<th>Information</th>
<th>Connection</th>
</tr>
</thead>
</table>
Demo

Initial Access:
From .hta to full Meterpreter
Questions?

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

@kmkz_security