Harness: Powershell Weaponization Made Easy (or at least easier)

Rich Kelley
@RGKelley5
What’s this all about anyway?

• Audience:
  • Penetration testers
  • Red Teams
  • Powershell activists
  • Python enthusiasts

• Bottom line
  • Powershell weaponization can be somewhat cumbersome
  • Hopefully I’ve made that a little easier with the Harness tool set
Who is this guy?

- Computer science background
- Prior US Air Force Communications Officer
- Network engineer, software developer, penetration tester
- Currently focused on application pen testing
- Mostly I enjoy writing obscure utilities
  - pyhashcat
  - Keyboard walk generators
Why should I care?

• “...Microsoft’s Post-Exploitation Language” - @obsuresec

• Defenders should be more aware of the damage attackers can do with Powershell alone

• We need more research into incident response related to malicious Powershell use
  • DEF CON 22 - Ryan Kazanciyan and Matt Hastings, Investigating PowerShell Attacks
Powershell weaponization problem?

“How do you get your [Powershell] scripts running on your target machines, and effectively get your results back?” - @harmj0y
Hasn’t this problem been solved?

• Yep, but I’m a developer. Why use someone else’s solution when I can write my own (I’m kidding...sort of)

• Previous solutions were not as seamless as I wanted
  • Step 1: Gain access
  • Step 2: ?????
  • Step 3: Use powershell
  • Step 4: Pwn all things!

• A couple of very cool new solutions have recently been released
RDP – Copy/Paste or Import-Module
Remote shell – call powershell.exe
Metasploit – exec_powershell
Metasploit – Interactive PS Payloads

```bash
msf exploit(handler) > run
[*] Started reverse SSL handler on 192.168.142.129:4444
[*] Starting the payload handler...
[*] Powershell session session 3 opened (192.168.142.129:4444 -> 192.168.142.128:49175)
at 2015-07-15 11:58:19 -8400

Windows PowerShell running as user User1 on WIN7-TARGET
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\User1\Desktop> ls

Directory: C:\Users\User1\Desktop

PS C:\Users\User1\Desktop> for($i=1; $i -le 10; $i++){$i}
1
2
3
4
5
6
7
8
9
10
PS C:\Users\User1\Desktop>
```
Cobalt Strike – Beacon
My Development Requirements

1. Fully interactive remote Powershell console with the same capabilities as the native Powershell.exe

2. Ability to seamlessly import modules across the wire
Demo Time!
Under the hood

• Payload Requirements
  • .NET 3.0+
  • Powershell 2.0
  • System.Management.Automation Assembly

• Tested on:
  • Windows 7
  • Window 8
  • Windows 8.1
  • Windows Server 2008 R2
  • Windows Server 2012
Under the hood

- **Listener/Framework Requirements**
  - Python 3.4
  - Asyncio
  - Linux
    - Tested on Kali

- **Why Python? Why not Ruby? Why not Metasploit?**
  - Mostly for the learning experience
  - I prefer Python to Ruby (calm down)
  - Should be simple enough to port to Metasploit module
Under the hood

**Payload**

while script not valid: accumulate
end

**Handler**

PS C:\> ls
ls
PS C:\> ls
Directory C:\
Mode: LastWriteTime
----- ---------------------
d---- 2/2/1015
Under the hood

**Payload**

- Inbound script → True
- while !rcvd close signal:
  - accumulate
- end
- Socket ← ps.BeginInvoke

**Handler**

- PS C:/> ^import-module script.ps1
- Socket ← byte stream
- PS C:/> ^import-module script.ps1
- Directory C:\
  - Mode: LastWriteTime
  - ----- ------------------
  - d---- 2/2/1015
Questions?