PowerShell: Malicious by Design

Samuel – Valcan_K

Who Am I

Red Teamer | Pentester

- Specialize in Adversary Emulation
- USAF
- Prior SysAdmin
- Twitter: @Valcan_K

Introduction

PowerShell Is Everywhere!

So, why PowerShell?

- PowerShell is present on most Windows based systems.
- It's also available on some Linux based Operating Systems.
- Admins use it on a daily basis, so why shouldn't we?

PowerShell Was Designed For Hackers

Why is that?

- It’s user friendly.
- Easy to write scripts and automate tasks.
- Allows you to: enumerate hosts/networks, move laterally, evade and bypass detections.
- You can leverage the power of the .NET framework.

Common Offensive

PowerShell Tools

Tools to know about.

- PowerSploit
- Empire
- Invoke-Obfuscation
- Etc.

Enumeration

Scanning

Enumerate Hosts & Networks with PowerShell!

- There are several different ways we can enumerate hosts & networks using PowerShell.
- Let's talk about common methods to enumerate users, groups, and installed software on a host.
- Then we'll discuss how we can go about enumerating a network.

Host Enumeration

Common Commands:

- Gathering Users:
  - Get-LocalUser (local use)
  - Get-LocalGroup (local use)
  - Get-WmiObject (local and remote use)
  - Or we can use → ADSI (local and remote use)

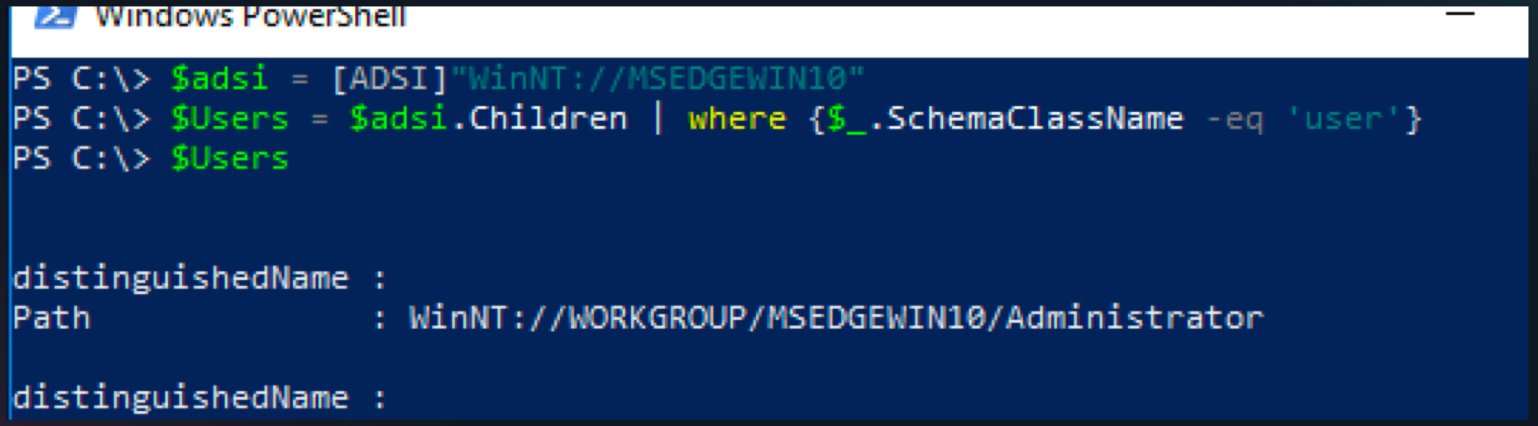
Get-LocalUser | Select * | Select Name

```
PS C:\> Get-LocalUser | select * | select name

Name
----
Administrator
DefaultAccount
Guest
IEUser
sshd
sshd_server
WDAGUtilityAccount
```

Get-WmiObject...

```
PS C:\> Get-WmiObject -ComputerName MSEDGEWIN10 -class win32_useraccount -filter "loca
laccount=true" | select pscomputername, name, status, accounttype

PSComputerName name               status   accounttype
-------------- ----               ------   -----------
MSEDGEWIN10    Administrator      Degraded         512
MSEDGEWIN10    DefaultAccount     Degraded         512
MSEDGEWIN10    Guest              Degraded         512
MSEDGEWIN10    IEUser             OK               512
MSEDGEWIN10    sshd               Degraded         512
MSEDGEWIN10    sshd_server        OK               512
MSEDGEWIN10    WDAGUtilityAccount Degraded         512
```

ADSI = Active Directory Service Interfaces



Host Enumeration

Common Commands:

- Gathering System Information:
  - Get-WmiObject
  - Get-NetTCPConnection
  - Get-Process

Get-WmiObject –Class Win32_Product

```
PS C:\Users\User> Get-WmiObject -class win32_product | Format-Table name

name
----
Microsoft .NET Framework 4.5.1 Multi-Targeting Pack
Universal CRT Headers Libraries and Sources
ClickOnce Bootstrapper Package for Microsoft .NET Framework
Windows SDK Redistributables
MSI Development Tools
Microsoft System CLR Types for SQL Server 2019 CTP2.2
Windows SDK Desktop Tools x86
```

Get-NetTCPConnection…

```
PS C:\Users\User> Get-NetTCPConnection -State Established

LocalAddress                        LocalPort RemoteAddress
------------                        --------- -------------
```

Get-Process

```
PS C:\Windows\system32> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessN
-------  ------    -----      -----     ------     --  -- --------
```

Network Enumeration

Common Commands:

- Enumerating Hosts, Users, and Groups:
  - Port Scanner with a 1 liner
  - Get-WmiObject (Get Users and Groups)
  - ADSI!

Port Scanner

```
PS C:\Users\Administrator> 1..1024 | % {echo ((new-object Net.Sockets.TcpClient).Connect("10.0.2.15",$_)) "Port $_ is open!"} 2>$null
Port 53 is open!
Port 80 is open!
Port 88 is open!
```

Get-WmiObject --- Users

```
PS C:\Users\Administrator> Get-WmiObject -Class ds_user -Namespace root\directory\ldap | Format-Table DS_sAMAccountName

DS_sAMAccountName
-----------------
Administrator
Guest
krbtgt
```

ADSI – Domain Users

```
PS C:\Users\Administrator> [ADSI]$users = "LDAP://CN=Users,DC=Research,DC=local"

$users.Children.Where({$_.schemaclassname -eq 'user'}) | select DistinguishedName, sAMAccountName,Name | Format-Table sAMAccountName,Name


sAMAccountName                                                                                  Name
--------------                                                                                  ----
{Administrator}                                                                                 {Administrator}
{bob.bobby}                                                                                     {bob bobby}
{Guest}                                                                                         {Guest}
```

ADSI – Domain Groups

```
PS C:\Users\Administrator> [ADSI]$groups = "LDAP://CN=Users,DC=Research,DC=local"

$groups.Children.Where({$_.schemaclassname -eq 'group'}) | select DistinguishedName, sAMAccountName,Name | Format-Table sAMAccountName,Name


sAMAccountName                                                  Name
--------------                                                  ----
{Allowed RODC Password Replication Group}                       {Allowed RODC Password Replication Group}
{Cert Publishers}                                               {Cert Publishers}
{Cloneable Domain Controllers}                                  {Cloneable Domain Controllers}
{Denied RODC Password Replication Group}                        {Denied RODC Password Replication Group}
{DnsAdmins}                                                     {DnsAdmins}
```

Lateral Movement

+ Remote Command Execution

Common Commands:

- Creating Remote Sessions, and Executing Remote Commands:
  - PSSession
  - Invoke-Command
  - Invoke-CimMethod

PSSession

```
PS C:\Users\Administrator> New-PSSession -ComputerName 10.0.0.2 -Credential User
```

Invoke-Command

```
PS C:\Users\Administrator> Invoke-Command -ScriptBlock {mkdir C:\Temp} -ComputerName Target1
```

Invoke-CimMethod

```
Invoke-CimMethod -ComputerName TestBox1 -ClassName Win32_Process -MethodName "Create" -Arguments @{ Path = "evil.exe" }
```

PowerShell and Evasion/Bypass Techniques

- There are several ways to go about using PowerShell for Evading and Bypassing Detections

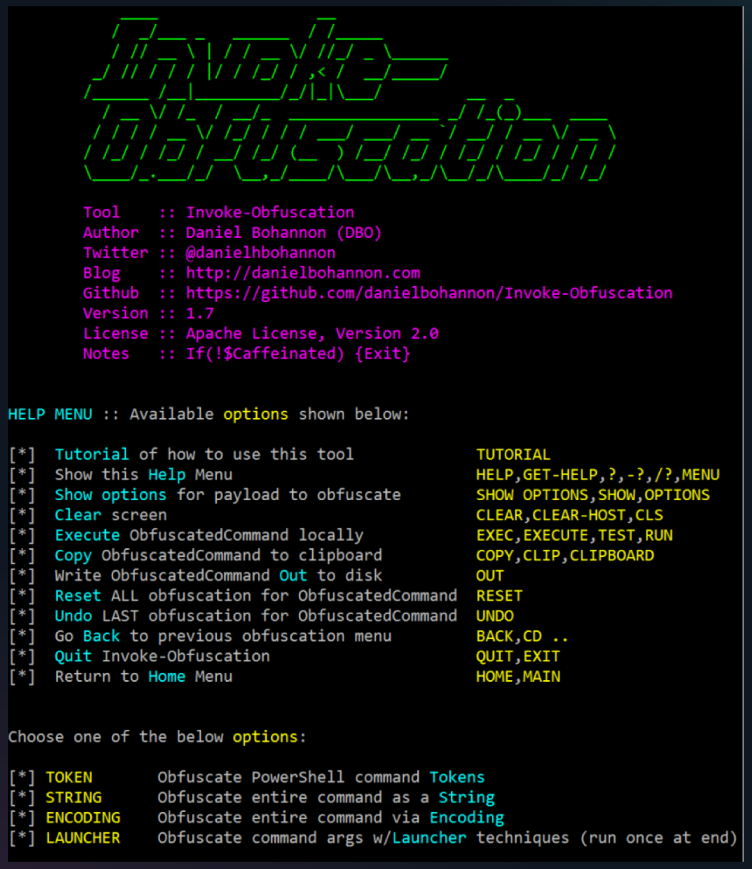
  A common method is the use of Obfuscation and/or Encoding.

  - Invoke-Obfuscation by Daniel Bohannon.

- Utilizing the .NET framework from PowerShell

  - To load binaries into memory.

Invoke-Obfuscation



Utilizing .NET

```
[System.Reflection.Assembly]::Load($YourData.ToArray())
[Test.Methods]::Main()
```

Using PowerShell to Load an Assembly/Binary

DEMO

The Loaded Code

```
using System;
using System.Net.Http;
using System.Threading.Tasks;


namespace DownloadPage

{
    0 references
    public class Methods
    {
        0 references
        public static async Task Main(string[] args)

        {
            using var client = new HttpClient();
            client.DefaultRequestHeaders.Add("User-Agent", "CobaltStrike");
            var content = await client.GetStringAsync("https://www.cobaltstrike.com/download");
            string filename = @"C:\temp\output.txt";

            System.IO.File.WriteAllText(filename, content);

            Console.WriteLine(content);
        }
    }
}
```

References

References:

- https://docs.microsoft.com/en-us/powershell/module/nettcpip/get-nettcpconnection?view=win10-ps
- https://www.powershellbros.com/adsi-searching-for-an-user-object-in-active-directory/
- https://www.petri.com/active-directory-powershell-adsi-ldap
- https://www.sans.org/blog/pen-test-poster-white-board-powershell-built-in-port-scanner/
- https://docs.microsoft.com/en-us/windows/win32/adsi/active-directory-service-interfaces-adsi
- https://docs.microsoft.com/en-us/powershell/module/nettcpip/get-nettcpconnection?view=win10-ps
- https://www.powershellbros.com/adsi-searching-for-an-user-object-in-active-directory/
- https://www.petri.com/active-directory-powershell-adsi-ldap
- https://www.sans.org/blog/pen-test-poster-white-board-powershell-built-in-port-scanner/
- https://ss64.com/ps/invoke-cimmethod.html
- https://github.com/danielbohannon/Invoke-Obfuscation
- https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon
- https://github.com/EmpireProject/Empire

Resources

Things to check out:

- Learn PowerShell: https://underthewire.tech/
- Learn Adversary PowerShell Tactics: https://github.com/specterops/at-ps
- BlackHat Talk by @retBandit: https://www.blackhat.com/docs/eu-17/materials/eu-17-Thompson-Red-Team-Techniques-For-Evading-Bypassing-And-Disabling-MS-Advanced-Threat-Protection-And-Advanced-Threat-Analytics.pdf

Spicy Fish Tacos & Margaritas

Image: simplyscratch.com/Laurie McNamara

Image: Liquor.com/Tim Nusog