PowerShell Hunting Cheat sheet

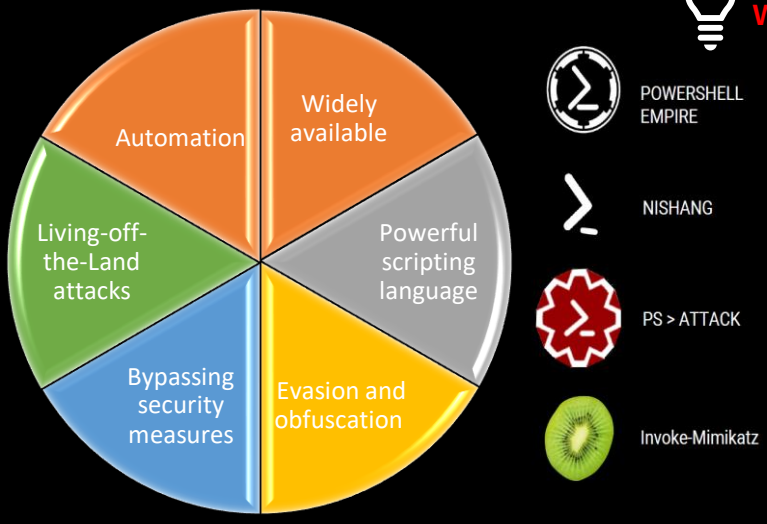
What is PowerShell?

- Task automation and configuration management framework from Microsoft.
- Consisting of a command-line shell and associated scripting language.
- Built on the .NET Framework.
- Enabling administrators to perform administrative tasks on both local and remote Windows systems

Basic Details of PowerShell

ID: T1086
Tactic: Execution
Platform: Windows
Permissions Required: User, Administrator
Data Sources: PowerShell logs, Loaded DLLs, DLL monitoring, Windows Registry, File monitoring, Process monitoring, Process command-line parameters
Supports Remote: Yes

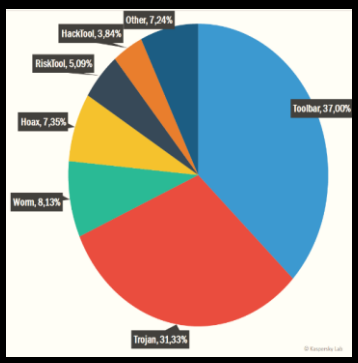
Why attackers love PowerShell ?



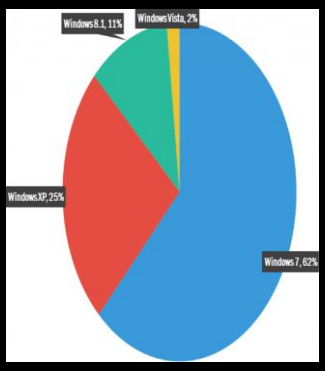
Well-known PowerShell Offensive Frameworks

- PowerShell Arsenal
- PowerShell-AD-Recon
- DSInternals
- DSCCompromise
- Inveigh
- Invoke-WMILM
- PS>Attack
- PowerSploit
- PowerCat
- Empire
- DarkObserver
- PowerMemory
- Invoke-Mimikatz
- Invoke-Mimikittenz
- Offensive-PowerShell
- Kautilya
- Nishang
- PoshRat
- PowerShell Suite
- OWA-Toolkit
- Sherlock
- Invoke-Phant0m

Some statics on PowerShell Attack



Rise of PowerShell Malware



Most attacked operating systems by malware written in any .NET compatible language.

Bypassing the PowerShell Execution Policy

Get-Content .\script.ps1 | powershell.exe –noprofile –

type .\script.ps1 | powershell.exe –noprofile –

powershell -command "Write-Host Hello from PowerShell!!!"

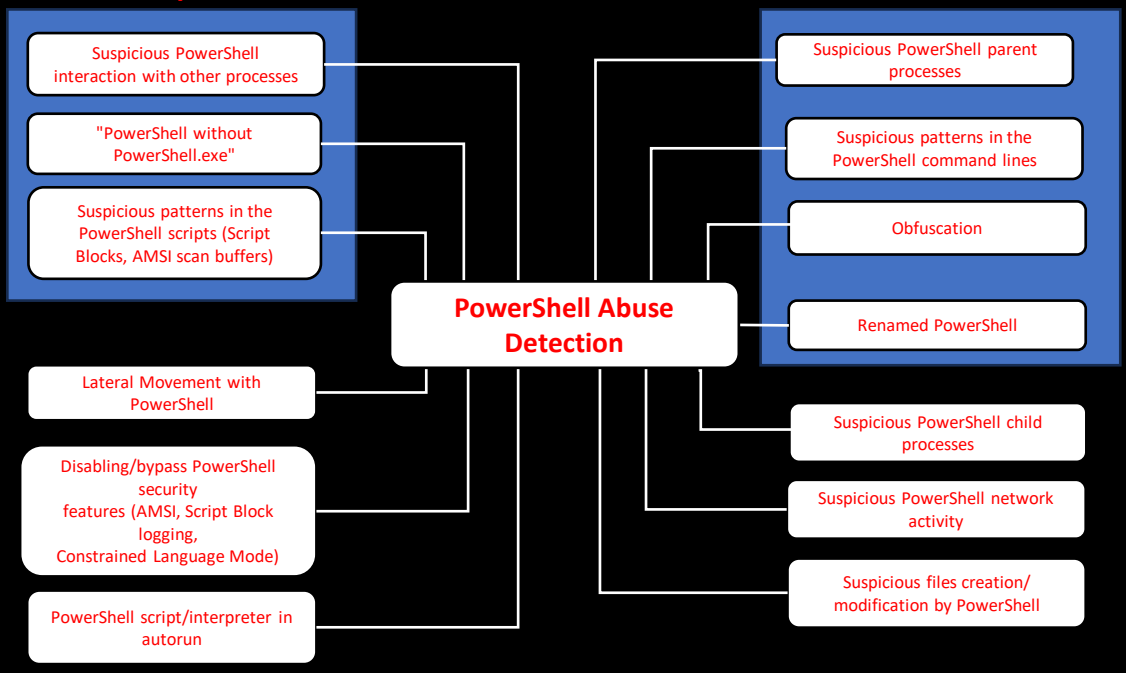
Invoke-Command –scriptblock {Write-Host Hello from PowerShell!!!}

Get-Content .\script.ps1 | Invoke-Expression

Set-ExecutionPolicy Bypass -Scope Process

powershell -ExecutionPolicy Bypass -File .runme.ps1

PowerShell abuse patterns



Check all Suspicious command for Autorun registry keys modification events:

```
if [winlog][channel] == "Microsoft-Windows-Sysmon/Operational" and [winlog][event_id] == 13 and
[winlog][event_data][RuleName] == "reg_persistence_cmdline" and [winlog][event_data][Details] != "" {
        mutate {
                add_field => { "[winlog][event_data][CommandLine]" => "%{[winlog][event_data][Details]}" }
        }
}
```

Check all Suspicious command for Command Line WMI consumers creation events:

```
if [winlog][channel] == "Microsoft-Windows-Sysmon/Operational" and [winlog][event_id] == 20 {
        if [winlog][event_data][Type] == "Command Line" and [winlog][event_data][Destination] != "" {
                mutate {
                        add_field => { "[winlog][event_data][CommandLine]" => "%{[winlog][event_data][Destination]}" }
                }
        }
}
```

Suspicious PowerShell parent process:

Parent process application category	Possible attack vector	Possible MITRE ATT&CK techniques
MS Office App / PDF Reader	Doc with macros/DDE etc., vulnerability exploitation	T1204: User Execution ; T1173: Dynamic Data Exchange ; T1203: Exploitation for Client Execution T1064: Scripting (macros)
MS Outlook	Persistence via Outlook, process execution via Outlook.Application COM	T1137: Office Application Startup TT175: Distributed Component Object Model
Internet Browser	Browser or plugin vulnerability exploitation	T1189: Drive-by Compromise T1203: Exploitation for Client Execution
Web Server	Web Shell, vulnerability exploitation	T1100: Web Shell ; T1210: Exploitation of Remote Services ; T1190: Exploit Public-Facing Application
MS SQL Server	xp_cmdshell, vulnerability exploitation	T1210: Exploitation of Remote Services T1190: Exploit Public-Facing Application
Other Server Applications	Vulnerability exploitation	T1210: Exploitation of Remote Services T1190: Exploit Public-Facing Application

Event sources for detection of PowerShell abuses

EID 4688 with command lines (Process Creation)
EID 4698 (Scheduled Task Creation)
EID 4702 (Scheduled Task Modification)

Windows Security Event Log

EDR

EID 5861
EID 5858

Windows WMI logging (Microsoft-Windows-WMI-Activity/Operational event log)

Windows System Event Log

EID 7045 (Service Creation)

EID 4104
EID 4103

Script Block / Module Logging (Microsoft-Windows-PowerShell/Operational event log)

PowerShell Engine state logging

EID 400

Event Sources to hunt for PowerShell Abuses

Process creation events
DLL loading events
Inter-process communication events (memory access, remote threads creation)
Process network activity events
Registry keys modification events
File creation/modification events

Sysmon/EDR

Transcription Logging

PowerShell ConsoleHost_history. txt

EPP (AV) detects log

Antimalware Scan Interface (AMSI/Operational event log)

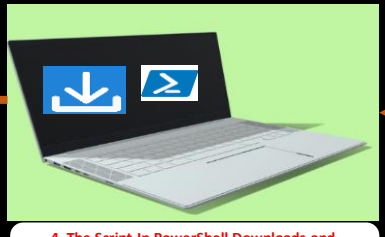
Behavior detects on PowerShell processes
Signature/heuristic detects on PowerShell scripts
Signature/heuristic detects on PowerShell scripts bodies, received via AMSI
Signature/heuristic detects on ConsoleHost_history.txt files

Fileless Attack Example Exploiting PowerShell

1. The User Visits an Infected Website or Opens a Malicious Link.

2. Flash Java Is Loaded on the Website and the User's Device Is Scanned for Vulnerabilities.

3. A Shell Code Launches PowerShell. The Attacker Can Run Malicious Command Line Operations In the User's OS Memory.



4. The Script In PowerShell Downloads and Executes a Payload Carrying Out Malicious Activity in the User's OS Memory.

5. The Fileless Attack Is Successful. The Cybercriminal Can Now Steal Sensitive Data, Launch a Ransomware Attack, and More.