



A fully configurable and extendable Bash obfuscation framework. This tool is intended to help both red team and blue team.

https://bashfuscator.readthedocs.io/en... #bash #linux #linux-shell #obfuscation #red-team #blue-team #obfuscation-framework #modular #evasion #incident-response

Table with 5 columns: 295 commits, 5 branches, 0 releases, 4 contributors, MIT license.

Branch: master, New pull request, Find file

Clone or download

Table of project files and their commit history, including bashfuscator, docs, img, scripts, test, .gitignore, CONTRIBUTING.md, LICENSE, README.md, readthedocs.yml, setup.py, time_measure.sh.



Bashfuscator

Documentation

What is Bashfuscator?

Bashfuscator is a modular and extendable Bash obfuscation framework written in Python 3. It provides numerous different ways of making Bash one-liners or scripts much more difficult to understand.

The purpose of this project is to give Red Team the ability to bypass static detections on a Linux system, and the knowledge and tools to write better Bash obfuscation techniques.

This framework was also developed with Blue Team in mind. With this framework, Blue Team can easily generate thousands of unique obfuscated scripts or commands to help create and test detections of Bash obfuscation.

Payload support

Though Bashfuscator does work on UNIX systems, many of the payloads it generates will not. This is because most UNIX systems use BSD style utilities, and Bashfuscator was built to work with GNU style utilities.

Installation & Requirements

Bashfuscator requires Python 3.6+.

On a Debian-based distro, run this command to install dependencies:

```
sudo apt-get update && sudo apt-get install python3 python3-pip python3-argcomplete xclip
```

On a RHEL-based distro, run this command to install dependencies:

```
sudo dnf update && sudo dnf install python3 python3-pip python3-argcomplete xclip
```

Then, run these commands to clone and install Bashfuscator:

```
git clone https://github.com/Bashfuscator/Bashfuscator
cd Bashfuscator
python3 setup.py install --user
```

Only Debian and RHEL based distros are supported. Bashfuscator has been tested working on some UNIX systems, but is not supported on those systems.

Example Usage

For simple usage, just pass the command you want to obfuscate with -c, or the script you want to obfuscate with -f.

```
$ bashfuscator -c "cat /etc/passwd"
[+] Mutators used: Token/ForCode -> Command/Reverse
[+] Payload:
${@/1+Jau/+<b=k } p""r""i""n""s""t\u0066' %s "${ %*%Fr\[47T2 } ${##@|j.g } "r""e""v <<< '

[+] Payload size: 1232 characters
```

You can copy the obfuscated payload to your clipboard with --clip, or write it to a file with -o.

For more advanced usage, use the --choose-mutators flag, and specify exactly what obfuscation modules, or Mutators, you want to use in what order. Use also the -s argument to control the level of obfuscation used.

```
bashfuscator -c "cat /etc/passwd" --choose-mutators token/special_char_only compress/bzip2 string/file_gl
[+] Payload:
"${@#b }" "e"$\166"a""${@}l "${ {@}m"'$k\144''ir -p '/tmp/w/'$*--);${\x70}'${@AZ}"rin""tf %
??"; prin${@#K. }tf %s 'wyg01ujRoaGhoNMgygAJNKSp+1Mgkx6pgcGRhDDRGMNDTQA0ABoAAZDQIkhCkyPNIm1DTQeppjRDTTQ
' "${@,,} " &&${*}pri""\n${*},}tf %s 'RELKWCoKqgFP5VELV55qmdRJqe1AziQTBBM9BbllyhIQN8VyrjiIRkd2LQIrwLY2E9
';"${@, }" ${\x70}'rintf %s 'c1dkczJBNSB1gA0sW2tAFoIhpWtL3K/n68Vvs4Pt++D6+2X4FILnaFw4xaW1bbaJbKJbGLou0j3
?' ; ${*/~} p""s${@#v1 }r1""n""tf %s ' pr""i""i""i""i""i""s""i""i""n""x74''''''f %s "${ prin${*/N/H }tf ""
' "${*};"p"rin"'s'\x74f' %s 'Gs02t3sw+yFjnPjcxLJ5I5XTnNzNMjJnSm0ChZQfSIFbxj6xzTfngZc4YbPvaCS3jMXvYinGLUW
?
&& "${@^ }" pr""intf %s 'Q+kXS+VgQ90k1AYb+q+GYQQz14xQD1AGRJBCQbaTS11cpkRmZ1hSkDjckn3UADEBeXJAIF1yES
???' ${*^}; ${@} "${@%I }"pri""nS'\x74f' %s '1w6xQDwJRXSpvdUvYXcku4JBC1340A''''''''''''''''''''''''''''''''''''''''''''''
??
' "${@#b } ; pr'i""ntf %s 'g8oZ91rJxesUWCiAwikkyQD1m3Zw341vr1i0kuGmu1Z2Q5IkkgYAAJFzgg1RwXergULhLMNTjch
?' "${@/#Y }" ; ${c\141it' '/tmp/w/'/???? ${*/m};"${@,, }" "${\162'\m '/tmp/w/'/???? &&${@ }rmd\ir '/t
```

For more detailed usage and examples, please refer to the documentation.

Extending the Framework

Adding new obfuscation methods to the framework is simple, as Bashfuscator was built to be a modular and extendable framework. Bashfuscator's backend does all the heavy lifting so you can focus on writing robust obfuscation methods.

Authors and Contributors

- Andrew LeFevre (capnspcehook): project lead and creator
Charity Barker (cpbarker): team member
Nathaniel Hatfield (343Church): writing the RotN Mutator
Elijah Barker (elijah-barker): writing the Hex Hash, Folder and File Glob Mutators
Sam Kreischer: the awesome logo

Credits

- danielbohannon, whose excellent Invoke-Obfuscation and Invoke-DOSfuscation projects gave capnspcehook the idea to start writing Bashfuscator, and insight on how to write robust obfuscation methods.
DissectMalware, whose tweets on Bash obfuscation formed the backbone of some Mutators, and provided ideas for other obfuscation techniques.
ConsciousHacker, whose insight and advice has helped the team greatly.
Bash logo was originally from https://github.com/odb/official-bash-logo.

Disclaimer

Bashfuscator was created for educational purposes only, use only on computers or networks you have explicit permission to do so. The Bashfuscator team is not responsible for any illegal or malicious acts preformed with this project.