Bashfuscator Documentation

Release 0.0.1

Bashfuscator Team

Feb 14, 2020

Contents

1 Introduction **3**
- 1.1 Mutator types . . . 3

2 Usage **5**
- 2.1 Basic CLI Usage . . . 5
- 2.2 Advanced CLI Usage . . . 7

3 Mutator Documentation **9**
- 3.1 Size and Time Ratings . . . 9
- 3.2 Index . . . 10

4 Development **23**
- 4.1 Adding New Mutators to The Framework . . . 23

5 Module Documentation **27**
- 5.1 common - Modules used all over the framework . . . 27
- 5.2 core - Modules core to the framework's functionality . . . 28

Python Module Index **41**

Index **43**

Contents:

CHAPTER 1

Introduction

Bashfuscator is built to be a modular, flexible Bash obfuscation framework. It achieves this by organizing different obfuscation techniques and methods into modules within the framework, called Mutators. Different obfuscation 'recipes' can be created by stacking different Mutators, giving the resulting payload varying characteristics and appearances.

1.1 Mutator types

There are 5 types of Mutators:

1. **Command Obfuscators**
    - Simple obfuscators that leverage behavior of commands or binaries present in a Linux environment
    - Obfuscates entire input in one chunk
2. **String Obfuscators**
    - Obfuscators that use more advanced features/binaries
    - Breaks input into chunks, obfuscates those chunks, then builds up input by concatenating standard output of all of the different obfuscated chunks
3. **Token Obfuscators**
    - Leverages Bash functionality or behavior to obfuscate commands
    - Typically don't use any external binaries
    - Obfuscates entire input in one chunk
4. **Encoders**
    - Encodes the entire input and decodes it using a stub.
5. **Compressors**
    - Compresses the input and decompresses it using a stub, using various compressors typically available in a Linux environment

CHAPTER 2

Usage

2.1 Basic CLI Usage

Before starting to obfuscate for the first time, it may be useful to have a list of all of the available Mutators contained in the Bashfuscator framework. The *-l* option will do just that, and give size and time ratings, descriptions, and more.

```
$ bashfuscator -l
Command Obfuscators:

Name: Case Swapper
Description: Flips the case of all alpha chars
Size Rating: 1
Time Rating: 1
File write: False
Author: capnspacehook

Stubs:

    Name: bash case swap expansion
    Binaries Used: None
    Size Rating: 1
    Time Rating: 1
[snip]

Token Obfuscators:

Name: ANSI-C Quote
Description: ANSI-C quotes a string
Size Rating: 3
Time Rating: 1
Binaries used: None
File write: False
Notes: Requires Bash 4.2 or above
Author: capnspacehook
```

(continues on next page)

(continued from previous page)

```
Credits: DissectMalware, https://twitter.com/DissectMalware/status/1023682809368653826

Name: Special Char Only
Description: Converts commands to only use special characters
Size Rating: 4
Time Rating: 2
Binaries used: cat
File write: False
Notes: Will break when run in Bash's debug mode. Also compresses extremely well
Author: capnspacehook
Credits: danielbohannon, https://github.com/danielbohannon/Invoke-Obfuscation
        Digital Trauma, https://codegolf.stackexchange.com/questions/22533/weirdest-
↪obfuscated-hello-world
[snip]
```

When you're ready to start obfuscating, use the *-c* or *-f* options to specify a one-liner or script file to obfuscate, and Bashfuscator will take care of the rest, randomly choosing Mutators to obfuscate the input with. Bashfuscator only requires one of those two options, although many more are available to fine-tune the obfuscation.

```
$ bashfuscator -c "cat /etc/passwd"
[+] Payload:

eval "$(printf %s '")enod;"}]Y5DFSN$[A55_NI{$" s% ftnirp od;6 3 2 2 7 5 9 8 1 0 9 4 1␣
↪7 8 ni Y5DFSN rof;)/\ c a d p  \ w s t e(=A55_NI($" lave'|rev)"

[+] Payload size: 149 characters
```

The *-s* and *-t* options control the added size and execution time of the obfuscated payload respectively. They both default to 2, but can be set to 1-3 to control the generated payload more closely. The higher the *-s* or *-t* options, the greater the variety of the payload, at the expense of added size. When a low setting for the *-s* or *-t* options is set, Bashfuscator will limit itself to using Mutators that increase the size and execution time of the payload slightly. With high values, Bashfuscator has a chance to pick any combination of it's Mutators!

When you've finally settled on an obfuscation recipe to use, two output options are available: *–clip* and *-o*. *–clip* will automagically copy the obfuscated payload to your clipboard, and *-o* will write the payload to a file that you specify.

You should make sure the obfuscated payload works as expected, and the *–test* option will make that easier. When used, *–test* will invoke the obfuscated payload in memory, and show the output. When *-o* is used to specify an output file to write to, the output file will be run after the payload is written to it.

```
$ bashfuscator -c "cat /etc/passwd" --test
[+] Payload:

eval "$(ijmduN3D=(\[ r f 5 4 G U \" a i s p 1 t \% \} \  e \) \/ \\ 0 b J k z 7 \] \;␣
↪\{ \| D \( X 2 h 3 \= 9 V 8 w n \$ B c 6 d o);for s7SQJyu8 in 11 1 9 42 13 2 16 14␣
↪10 16 7 43 32 24 44 39 8 6 33 37 32 20 19 16 45 16 10 16 47 16 41 16 13 16 11 16 17␣
↪16 8 16 20 16 18 28 2 48 1 16 31 25 35 24 23 36 41 5 16 9 42 16 12 16 40 16 3 16 38␣
↪16 21 16 26 16 3 16 12 16 21 16 46 16 40 16 34 16 34 16 4 16 36 28 47 48 16 11 1 9␣
↪42 13 2 16 14 10 16 7 43 29 24 44 39 8 6 33 0 43 31 25 35 24 23 36 41 5 27 15 7 28␣
↪47 48 42 17 18 7 30 22 8 10 35;do printf %s "${ijmduN3D[$s7SQJyu8]}";done)"

[+] Payload size: 578 characters
[+] Testing payload:

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

(continues on next page)

(continued from previous page)

```
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
[snip]
```

2.2 Advanced CLI Usage

Note: All options are tab-completable! This makes CLI usage sooo much easier, especially when using long options.

The *–layers* option will control the amount of obfuscation layers Bashfuscator will apply to the input. The default is 2 layers. This is useful to control the amount of obfuscation applied to the input.

The *–full-ascii-strings* option is an interesting one. When used, the full ASCII character set is used when randomly generating strings to be used within the final payload. This means non-printable characters can possibly exist within the obfuscated payload, potentially (hopefully) messing with tools and regex used to examine your payload.

You can further fine-tune the obfuscation process by using the *–choose-mutators* option. This option allows you to manually select which Mutators Bashfuscator will use, and in what order. You can create some pretty creative obfuscation recipes using this option. If used with the *–layer* option, the obfuscation recipe can be layered upon itself multiple times.

```
$ bashfuscator -c "cat /etc/passwd" --choose-mutators token/special_char_only␣
↪compress/bzip2 command/case_swapper command/reverse --layers 2 --test
[+] Payload:

printf %s "$(rev <<<'hsab|")"}~~BiT8Vs5Q{$" s% ftnirp;'"'"'HSAB|")C- 2PIZNUB|D-␣
↪46ESAB|GPgu/3qcfofj3f/
↪NZi1eLAfCjO8uB+++Xvp8RH+9+FFgauqrDLKMXJsr4xvIOpVD++89yEDpN1j9tHK5oGIcIoOgkisz1ImrbrWbaQikXLOGO0SMKo
↪wjFHVBHpNqx4JH3qvSeXHDNvtZ5YJ2YGRrtZQ1YO5YMY9Eo2c7z3UzYNbh3zUt62yoFLLwc9kcTS+GToSKM1R7MOvA6IOtXsDcl
↪VV+eFCYuYfidsiHLmvvPQkkvLkVeUV99/4t/HIO6+JF+Tpuev5/
↪051COOG1rvGtprv1WdKuh8nKaWOeaiIaXf7beyGyfsei9G22AZcEmUmo+iNF1+e+rLA/
↪3Zt7FV8P9UN9Q73SN5NdFidLYRjELTp5yOsO2VsqMqNVgXQ1YqruWDR/0dF6tOxzRVU+K+IO+GpFeb+O/
↪45jQ91ep9QzIVBwqRKzwMCmuses4HwkwRXwae1DZKB13uhj0mEwxBk78vWIiU8NQ8tc6dLd8jxG5u+axFb/
↪3FjjHKKaFp/N3bqxrHeHKmDw+7PNi5+Ks2NOFJIYfOUikS6HQ6olmpeOExLQWRTl/6pVIuUNEvn8c/
↪1R+Thb7IKrP8EpbF8Dz+4Rz8zTWHiQqj1QgEvBW8UvnIVAfUZ06Liisefxate WsDUT6ljz7tAFtJ3RrTvKapsajlcbacHefutSr
↪fgcJ50S4sxIsdFnbTGpeqLC/
↪s2njiIEd2Q0PVnkuTMBgp7lV6sL9wVvnvblvbbacamQivfdvS133vVRRzY9HasvnGIf32yUoLuxxYLQ4CZ20XjLgxK4CBzgZ2CU
↪xplrvmreHcsbJksvSefBPR5DXzwJJdEKx+glQBd5EMPRdTOGSDo+COOgF067HWBepUnoRorUnChhgnyv5KcjxNqfh1d4BeOdBUl
↪M3ZetPgvYrieVUVdjaHejazKa1tZ46Y1Xn8gwiFKVMNNrQ3HP5SX5dLWm048kdzAtB85kFr1y0JNMSeK1vU5s9IhhEN7DQcwQOu
↪aX8mO8KenP5XdGqaBtObLqigP8dHaVaX8d2gtEGbGFAWKN51emb4NMGjX208OHNbBBmHymqCu+Fm/
↪MFBAOTTXKM0MGTz+16/3qBBJfVx95j9Q5558WN8PtJbufHUJa1OiyAlH/
↪swrhvwswUDlR1HkVUlrZuU+i78SFEGYZ6JQI/
↪XbSV8LFEKp0pFOiQ2ZzMR11CMDorXvnxQVX35BEQCTeBrEm9+WEvu0+1rTR4rIVzXynZmZOPtV09TVUG08SN9+wYDdzbrLUen0X
↪K3Z1SxNS5X3+4ZL3rhikf0k35oJhfLj7QdCWMHJRYqw9wMbNVKczv9CGg7C9EddoyY9Iz60ZPdxu49zJnwyI71vhpKBM9+o1kPr
↪itdwQBokRMkXmJgFYCoL4hUzVloS9Ox3woisp2TqJWSDPKVx6SMqKXZyQgKwnPZW7r9iVv7lCBrzGNvRbUjo50qDMbNkBjES3mC
↪T+y7p+rp6T+6lEgF9RBDs32VYJsr5fHkH/
↪fpQna1daGgqt6bn0KIAQKc0O6hjpPn1JAJYtmfBOYmPeeHeHrtdbmnnIHmHjmrPFQsLG+PgGAd0Kr9qM0gnftEOHg1tKObAAaaA
↪9b+V39gG/XyXayQb1/DwtzswbftoOPLq FTNIRP($" S% FTNIRP'"'"'=BiT8Vs5Q($" s% ftnirp')
↪"|bash

[+] Payload size: 5810 characters
```

(continues on next page)

(continued from previous page)

```
[+] Testing payload:

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
[snip]
```

Tab-completion is especially helpful when using the *–choose-mutators* option.

```
$ bashfuscator -c "cat /etc/passwd" --choose-mutators
command/case_swapper     compress/bzip2           encode/base64            string/
↪file_glob         string/forcode           token/ansi-c_quote
command/reverse          compress/gzip            encode/urlencode         string/
↪folder_glob       string/hex_hash          token/special_char_only
$ bashfuscator -c "cat /etc/passwd" --choose-mutators string/
string/file_glob    string/folder_glob  string/forcode      string/hex_hash
$ bashfuscator -c "cat /etc/passwd" --choose-mutators string/
```

CHAPTER 3

Mutator Documentation

3.1 Size and Time Ratings

Bashfuscator has a size and time rating system to help users find obfuscators that work best with their goals. For sandbox evasion, for example, a runtime increasing obfuscator may be useful. The size and time rating system is also designed to help developers benchmark their obfuscators. The size and time rating systems use Big O notation to break mutators into categories depending on their size and time increase rates. The time_measure.py script is used to generate test data and automate statistical analysis of the results. The script runs obfuscated and unobfuscated versions of the same command and measures differences in size and execution time.

3.1.1 Testing Methodology:

In order to remove as much system dependent execution time as possible, the unobfuscated input command is ":;" repeated to form "commands" of different lengths. (":" is a bash alias for true.) This command interacts minimally with the testing system.

In order to generate reliable data, several precautions were taken. For formal testing, each mutator was run 10 times at each data point and the average values are plotted to the graph. To get a larger snapshot of how the mutator behaves, commands from length 1 to length 10,000 were tested. Datapoints are linearly spaced.

3.1.2 Size Ratings

The size rating system rates mutators based on the number of characters added to the original command once it has been obfuscated.

Size Ratings Table:

Size Rating	Obfuscator
1: O(1), Little to no size increase	Case Swapper, Reverse
2: O(log(n)), Logarithmic growth rate	Folder Glob, File Glob
3: O(5n), Linnear with constant of 5	Base 64
4: O(kn), Linnear with constant k > 5	Hex Hash, Special Char Only
5: O(n^2), exponential or worse	

Example:

The String/HexHash mutator adds a constant number of characters for every character in the original command, so it has a linnear growth rate. Because HexHash adds a relatively large amount of additional characters, it is given the "Large Linnear" growth rate and assigned the size rating of 4.

3.1.3 Time Ratings

The time rating system rates mutators based on the runtime increase caused by a mutator.

Time Ratings Table

3.2 Index

3.2.1 Command Obfuscators

Command/Case Swapper

Examples

Without Mangling

```
eval "$(E3LFbgu='CAT /ETC/PASSWD';printf %s "${E3LFbgu~~}")"
```

With Mangling

```
$@  "${@~~  }"  \p${@,,  }r""i${*,  }n${*%%+Y,B;pv }t${@%%@cil:\}W  }f  %s  "$(  _
↪aWJ5_a='CAT /ETC/PASSWD'  "${@%%B6qv#j@=  }"  ${*/+Re\ew?  }  &&  ${@%^JjcVY:I }  ␣
↪p\r"i"${*}n't'f %s "${_aWJ5_a~~}"  ${@%%Vm\)?X }  $@  ${*//^PNN  }  )"  $
↪{@//6?*G.\)\/LJ\[k3 }  | "${@~~  }"  $* $'\x62''a'sh  ${*/9\[>f  }  ${*~  }
```

Description

Case Swapper is an extremely simple Mutator that leverages one of Bash's parameter expansions to flip the case of any alphabetic characters of it's input. See https://wiki.bash-hackers.org/syntax/pe#case_modification for more information on the parameter expansion used.

Side Effects

None.

Detection

Case Swapper can be detected by looking for weird casings of normal commands, such as *ECHO* or *PRINTF*.

Dependencies

None. Not sure when Bash added support for case modification parameter expansions, but only ancient versions of Bash might not work with this Mutator.

Runtime Graph

The runtime graph has a strange shape, but it's still clear that Case Swapper adds very little runtime overhead to generated payloads.

Size Graph

As expected, Case Swapper hardly adds any extra size to generated payloads.

Command/Reverse

Examples

Without Mangling

```
eval "$(rev <<<'dwssap/cte/ tac')"
```

With Mangling

```
${*,  } ${*%%$<\(5 } e${*##\)@gLH\[@\)  }val  "$(    ${@^ }  "$@"  \p\r\intf   %s
↪'dwssap/cte/ tac'   ${@%lk_z|?   }   | ${@##V8x2rVp  }   ''r'e'""'v'  ${*#O<-.   } ␣
↪ "${@^^ }"   ;  ${*,,   } ${!@}    )"   "${@/LLN\}<~ }"
```

Description

Reverse is another simple Mutator. Quite simply, it reverses it's input. That's it.

Side Effects

None.

Detection

Look for common commands or strings reversed, such as *ftnirp* or *lave*.

Dependencies

The *rev* command, available in the *util-linux* package which is installed by default on both Debian and Fedora based systems. http://man7.org/linux/man-pages/man1/rev.1.html

Runtime Graph

Similar to Case Swapper. Adds almost no overhead to generated payloads.

Size Graph

Adds on average 30 characters to the input payload. Very minimal size increase.

3.2.2 String Obfuscators

String/File Glob

Examples

Without Mangling

```
eval "$(mkdir -p '/tmp/:&$NiA';printf %s 'a'>'/tmp/:&$NiA/
?
?';printf %s '/'>'/tmp/:&$NiA/
???';printf %s 'p'>'/tmp/:&$NiA/
??
';printf %s 't'>'/tmp/:&$NiA/?

?';printf %s 'd'>'/tmp/:&$NiA/


?';printf %s 'c'>'/tmp/:&$NiA/????';printf %s 's'>'/tmp/:&$NiA/
```

(continues on next page)

(continued from previous page)

```
??';printf %s '/'>'/tmp/:&$NiA/?
??';printf %s 'c'>'/tmp/:&$NiA/?


';printf %s 'w'>'/tmp/:&$NiA/

?
';printf %s 'e'>'/tmp/:&$NiA/?
?
';printf %s ' '>'/tmp/:&$NiA/??

';printf %s 'a'>'/tmp/:&$NiA/???
';printf %s 't'>'/tmp/:&$NiA/??
?';printf %s 's'>'/tmp/:&$NiA/
?

';cat '/tmp/:&$NiA'/????;rm '/tmp/:&$NiA'/????;rmdir '/tmp/:&$NiA';)"
```

With Mangling

```
  ${*//g\[JF#\)~/NozbEY}   ""ev\a''\l   "$(   ${@##nHu5D\}}  $'m\153'"dir -p   '/tmp/
↪v1[CZc'   "${@%%0zgA}" ${*//O:E:xX/h_,y67}   ;  "${@#:,f&N1\(}" ""p$'\u0072'""$'\x69
↪'""nt\f   %s   'c' >  '/tmp/v1[CZc/?


' ${@^^} ${*%%f\!H5W;fB} ;  ${@//xl;\);&5g}   $'p\u0072'"i"nt${*/T|HP/ZnD*}f   %s  '/
↪'  >   '/tmp/v1[CZc/
???'   ${@~} &&   ${*##ez^NO?} pr\i$'\156't"f"   %s  'c'   >   '/tmp/v1[CZc/????'   $
↪{@/3UXC}   ${*} ;  ${*~~}   p\r${*/hq\)C./9MfTInx}i${*##h,@<q}n''tf  %s 't' > '/tmp/
↪v1[CZc/?

?' ${@^} &&   ${*}  p'r'i$'\156'"t"f   %s 'p'  >   '/tmp/v1[CZc/
??
' ${*}  &&   ${!*} pr$'i\u006e'\tf %s   '/' >   '/tmp/v1[CZc/?
??'   ${!@} "${@,}"  && "${@##EBDf*QG}" ''pr"${@,}"i$'n\x74f'   %s  'a'  > '/tmp/
↪v1[CZc/
?
?' ${@//O$5\]NN/VVaUY} ${@^^}  &&  "${@//u2ad|7/_6=Q}" ${*,,}  \pr"i"'n'$'t\146'  %s
↪'s'   > '/tmp/v1[CZc/
?

'   "${@^}" &&  ${*}   ${!@}   ""\p""\ri\n''\t$'\u0066' %s   ' '   >   '/tmp/v1[CZc/??

'   ${*#\)hVX} ${@} &&   ${@^}   $'\u0070'"${@~~}"r''\i$'\156'${*#mYyniC^}t'f' %s 'e'␣
↪ > '/tmp/v1[CZc/?
?
'  "${@,,}" ${*//y%Udq} && $@  "${@#M\`20}"  "p"''\r$'\x69'\n''$'\164'"${@~~}"f %s 't
↪'  >   '/tmp/v1[CZc/??
?' ${*//q,Ngc/QT$nFQ}   "$@" && ${@%%v~4EwD.} ${*^}   \pri\n\t""f %s   'a' >  '/tmp/
↪v1[CZc/???
'   ${*,}  ${@^}  ; ${*//\`e&SC\[mR/4gpoZ<}   ${@%%hSEpI} 'p'"r"""\i"${@#eP<_Ic}"nt${*
↪%Wlj2&}f   %s 'd'   >   '/tmp/v1[CZc/
```

(continues on next page)

(continued from previous page)

```
?'   ${*##+#%cSv&R} &&   ${@}  ${*}p$'\u0072i'\ntf   %s  'w'   >   '/tmp/v1[CZc/

?
'   ${@//$\}\)\{D.W/mFM9sVWy}  "${@}" && ${*} "$@" p\r''"i"'n'\t\f   %s  's' >  '/tmp/
↪v1[CZc/

??'   ${*/Z*J.7/;%Sr-=} ;  "$@" ${*//tWz%fyO}   c""at '/tmp/v1[CZc'/???? ${*%tDVp8\!}␣
↪&&  ${@%%ETGt}   \r${*//W*\)c>/\{g8>\{}m   '/tmp/v1[CZc'/????  ${*^^}   ${*%%\}L*\}}
↪ ; "${@/~%SU|}"  ${*%%=Xvy:~ti}  r$'m\144i'"r" '/tmp/v1[CZc';  "${@/YvBZKc}"  )"  "$
↪{@,,}" ${@}
```

Description

This mutator randomly arranges file writes. Then, using bash file globbing, it cats all the files which are in the directory combining them in order.

Side Effects

- Causes lots of file writes for large payloads which could eventually kill SSDs unless run in a ramdisk.
- Easy to reverse engineer through dynamic analysis. Simply prevent the rm command, then cat * in the directory to reveal original bash code.

Detection

Dependencies

- cat, mkdir, and rm binaries (in coreutils) http://man7.org/linux/man-pages/man1/cat.1.html
- A writeable directory.

Runtime Graph

Size Graph

String/Folder Glob

Examples

Without Mangling

```
eval "$(mkdir -p '/tmp/)6fv8H/Qjy0I';printf %s 'c'>'/tmp/)6fv8H/Qjy0I/?';cat '/tmp/
↪)6fv8H/Qjy0I'/?;rm '/tmp/)6fv8H/Qjy0I'/?;rmdir '/tmp/)6fv8H/Qjy0I';mkdir -p '/tmp/
↪)6fv8H/[|,9X,';printf %s 'a'>'/tmp/)6fv8H/[|,9X,/?';cat '/tmp/)6fv8H/[|,9X,'/?;rm '/
↪tmp/)6fv8H/[|,9X,'/?;rmdir '/tmp/)6fv8H/[|,9X,';mkdir -p '/tmp/)6fv8H/19m7VV=';
↪printf %s 't'>'/tmp/)6fv8H/19m7VV=/?';cat '/tmp/)6fv8H/19m7VV='/?;rm '/tmp/)6fv8H/
↪19m7VV='/?;rmdir '/tmp/)6fv8H/19m7VV=';mkdir -p '/tmp/)6fv8H/<sp?S;pl';printf %s '
↪'>'/tmp/)6fv8H/<sp?S;pl/?';cat '/tmp/)6fv8H/<sp?S;pl'/?;rm '/tmp/)6fv8H/<sp?S;pl'/?;
↪rmdir '/tmp/)6fv8H/<sp?S;pl';mkdir -p '/tmp/)6fv8H/y}[a&e*';printf %s '/'>'/tmp/
↪)6fv8H/y}[a&e*/?';cat '/tmp/)6fv8H/y}[a&e*'/?;rm '/tmp/)6fv8H/y}[a&e*'/?;rmdir '/
↪tmp/)6fv8H/y}[a&e*';mkdir -p '/tmp/)6fv8H/m0fBUD`';printf %s 'e'>'/tmp/)6fv8H/
↪m0fBUD`/?';cat '/tmp/)6fv8H/m0fBUD`'/?;rm '/tmp/)6fv8H/m0fBUD`'/?;rmdir '/tmp/
↪)6fv8H/m0fBUD`';mkdir -p '/tmp/)6fv8H/cTiI7k[_';printf %s 't'>'/tmp/)6fv8H/cTiI7k[_/
↪?';cat '/tmp/)6fv8H/cTiI7k[_'/?;rm '/tmp/)6fv8H/cTiI7k[_'/?;rmdir '/tmp/)6fv8H/
↪cTiI7k[_';mkdir -p '/tmp/)6fv8H/GSUz&S(<';printf %s 'c'>'/tmp/)6fv8H/GSUz&S(</?';
↪cat '/tmp/)6fv8H/GSUz&S(<'/?;rm '/tmp/)6fv8H/GSUz&S(<'/?;rmdir '/tmp/)6fv8H/GSUz&S(<
↪';mkdir -p '/tmp/)6fv8H/b5Vlj`';printf %s '/'>'/tmp/)6fv8H/b5Vlj`/?';cat '/tmp/
↪)6fv8H/b5Vlj`'/?;rm '/tmp/)6fv8H/b5Vlj`'/?;rmdir '/tmp/)6fv8H/b5Vlj`';mkdir -p '/
↪tmp/)6fv8H/e2@Dx|';printf %s 'p'>'/tmp/)6fv8H/e2@Dx|/?';cat '/tmp/)6fv8H/e2@Dx|'/?;
↪rm '/tmp/)6fv8H/e2@Dx|'/?;rmdir '/tmp/)6fv8H/e2@Dx|';mkdir -p '/tmp/)6fv8H/v`o#X';
↪printf %s 'a'>'/tmp/)6fv8H/v`o#X/?';cat '/tmp/)6fv8H/v`o#X'/?;rm '/tmp/)6fv8H/v`o#X
↪'/?;rmdir '/tmp/)6fv8H/v`o#X';mkdir -p '/tmp/)6fv8H/)"M;u';printf %s 's'>'/tmp/
↪)6fv8H/)"M;u/?';cat '/tmp/)6fv8H/)"M;u'/?;rm '/tmp/)6fv8H/)"M;u'/?;rmdir '/tmp/
↪)6fv8H/)"M;u';mkdir -p '/tmp/)6fv8H/1`}KA';printf %s 's'>'/tmp/)6fv8H/1`}KA/?';cat
↪'/tmp/)6fv8H/1`}KA'/?;rm '/tmp/)6fv8H/1`}KA'/?;rmdir '/tmp/)6fv8H/1`}KA';mkdir -p '/
↪tmp/)6fv8H/[<}V';printf %s 'w'>'/tmp/)6fv8H/[<}V/?';cat '/tmp/)6fv8H/[<}V'/?;rm '/
↪tmp/)6fv8H/[<}V'/?;rmdir '/tmp/)6fv8H/[<}V';mkdir -p '/tmp/)6fv8H/K&X0[Fc';printf
↪%s 'd'>'/tmp/)6fv8H/K&X0[Fc/?';cat '/tmp/)6fv8H/K&X0[Fc'/?;rm '/tmp/)6fv8H/K&X0[Fc'/
↪?;rmdir '/tmp/)6fv8H/K&X0[Fc';rmdir '/tmp/)6fv8H';)"
```

With Mangling

```
${*/^7r\[^} ${*^} 'p'$'r\151'$'\x6e'\t\f %s "$( ${!*} "$@"m\k\dir -p '/tmp/
↪d`gDU$3/yO}^91g' ${@#2=#CA} && ${*##=DL_} "${@%%#=@jv}" p"r"in"${@,,}"tf %s 'c
↪' > '/tmp/d`gDU$3/yO}^91g/?' ${@%%*n:hA~o} ; ${*^} c"${@%%O4V4K}"a\t '/tmp/
↪d`gDU$3/yO}^91g'/? "${@%\{zD=eCn}" && "${@/\}i$<2M@/G=e_#xj}" ''r\m '/tmp/d`gDU
↪$3/yO}^91g'/? ${*/n\)iJU>X\!/H\`\]hK} && ${*#b:Z\)Kz} r""\m$'\u0064'${!*}ir '/tmp/
↪d`gDU$3/yO}^91g'; ${*^} "m"$'\153'd"i"$'\162' -p '/tmp/d`gDU$3/aZU=I1C' "${@^}"␣
↪ ; "${@//\{Oz\[\]t\[/u.Y\{\{2}" ''${*}pr"i"${@/u\(b3?nf;}ntf %s 'a' > '/tmp/
↪d`gDU$3/aZU=I1C/?' ${*%%FMuaZE_} ; ${!@} cat '/tmp/d`gDU$3/aZU=I1C'/? ${*/
↪h8.b} && ${*//INHA2/\(J\]fR} ${@,} r""\m '/tmp/d`gDU$3/aZU=I1C'/? ${@//\`\]\!se;
↪} "${@^^}" && ${@/ewEO5dn/0~1Z4} ''"${@%%alO%}"r"m"di\r '/tmp/d`gDU$3/aZU=I1C
↪'; ${@/SWlvS} ${@~} $'m\x6bd'${*#q3$.Jop}ir -p '/tmp/d`gDU$3/dvgMj@sB' $@ "${@
↪%o\(fZP=\]j}" ; ${*^} ${@~} "${@#\}wR\}&}"p$'\162'$'\151'n$'\x74'f %s 't' > ␣
↪ '/tmp/d`gDU$3/dvgMj@sB/?' "${@^^}" && ${*##yXr\`2} "c"a't' '/tmp/d`gDU$3/
↪dvgMj@sB'/? ${*^^} && ${*^^} $'\162'\m '/tmp/d`gDU$3/dvgMj@sB'/? ${@^^} && $
↪{*^^} r"m"$'d\u0069'''r '/tmp/d`gDU$3/dvgMj@sB'; ${*/0NnD+9} mkdir -p '/tmp/
↪d`gDU$3/"t}M*' ${*~~} && ${@//>bOU\(/%Czn%T} p${*/D|y\`}r$'i\156't"f" %s ' ' > ␣
↪ '/tmp/d`gDU$3/"t}M*/?' ${*//dNvZ} $* ; ${*^} ${*/m\"8YG/46o#VP\)c} $'\u0063'$
↪{*%|<m\`nf\[}a"t" '/tmp/d`gDU$3/"t}M*'/? "${@//MpAP50/k=qU3N}" && "${@##U~xru}" r
↪'''m' '/tmp/d`gDU$3/"t}M*'/? ${@~~} && ${*##\(tWms*O} ${*} rmd"${@%%\8sW*vJ>}
↪"i$'\x72' '/tmp/d`gDU$3/"t}M*'&& "${@/7VVx}" ${*} mk\d${*~}i\r -p '/tmp/d`gDU
↪$3/2-j:EUt' ${*^^} && ${*//*x$G&u1/0EEB-lNb} "${@%V\!#708D}" \p\ri''"${@#a-
↪fnLs\(}"n\tf %s '/' > '/tmp/d`gDU$3/2-j:EUt/?' ${*%%pwWr} && ${!*} ${@~~} $
↪'\u0063'\at '/tmp/d`gDU$3/2-j:EUt'/? "${@#~vu.|1\[V}" ${*^} ; $* r"m"''/tmp/
↪d`gDU$3/2-j:EUt'/? ${*//K\"m<a/,m$Dr} && ${!@} ${!*} ""r${*##&:kF91u}mdir '/tmp/
↪d`gDU$3/2-j:EUt'; "${@}" mk"""${@/9vNj/a\"3M=D}"di'r' -p '/tmp/d`gDU$3/.t!5B'␣
↪ {@N3RT>} "$@" && ${*//*lF3/@<3A5\=h} ${*^} \p$'\u0072i''n'$*t'f' %s ␣
↪ 'e' > '/tmp/d`gDU$3/.t!5B/?' ${@^} && $* ${*//\`\}Ox} c''${*^^}a''t '/tmp/
↪d`gDU$3/.t!5B'/? ${@%%kxHp9v2@} ${*^} ; "${@#^=2Y==}" $'\162m' '/tmp/d`gDU$3/.
↪t!5B'/? ${*##xz6f5d3$} ${@##fXK@} ; "${@,,}" ${*~~} $'\u0072'm${@^}d''${@~}i$
↪{*/e7.FHZY}r '/tmp/d`gDU$3/.t!5B'&& "${@~}" ${@} mk''d${*^}i''$*r -p '/tmp/
```

(continues on next page)

(continued from previous page)

Description

Like file_glob, but organized into folders to make it harder to reverse engineer. If you prevent the rm command, then cat * in the subdirectories will reveal the original bash code, but it will be somewhat scrambled.

Side Effects

- Causes lots of file writes for large payloads which could eventually kill SSDs unless run in a ramdisk.
- Depending on size setting, parts of the original source

Detection

Dependencies

- cat, mkdir, and rm binaries (in coreutils) http://man7.org/linux/man-pages/man1/cat.1.html
- A writeable directory.

Runtime Graph

Size Graph

String/Hex Hash

Examples

Without Mangling

```
eval "$(printf "\x$(printf %s 'vceT'|md5sum|cut -b 28-29)";printf "\x$(printf %s '\dW,
↪'|md5sum|cut -b 30-31)";printf "\x$(printf %s 'w.~%!"'|md5sum|cut -b 24-25)";printf
↪"\x$(printf %s 'NDU}m^,z'|md5sum|cut -b 30-31)";printf "\x$(printf %s 'OB=40
↪'|md5sum|cut -b 9-10)";printf "\x$(printf %s 'F<0xpV8'|md5sum|cut -b 16-17)";printf
↪"\x$(printf %s '*<[`^b'|md5sum|cut -b 19-20)";printf "\x$(printf %s 'Ap5n]t,\
↪'|md5sum|cut -b 22-23)";printf "\x$(printf %s 'Qtll-'|md5sum|cut -b 13-14)";printf
↪"\x$(printf %s ':X],G'|md5sum|cut -b 15-16)";printf "\x$(printf %s '>^A5
↪'|md5sum|cut -b 8-9)";printf "\x$(printf %s '8<E2O'|md5sum|cut -b 22-23)";printf "\x
↪$(printf %s 'l>e@MtL4'|md5sum|cut -b 17-18)";printf "\x$(printf %s 'u"CmV
↪'|md5sum|cut -b 9-10)";printf "\x$(printf %s '{Z<3n9'|md5sum|cut -b 18-19)")"
```

With Mangling

```
"${@##cW^5Z}"  eva'l' "$(  ${*//Z;\{sMUgh/,75Oq}   ''pri"n"'t'f "\x$(   "${@//@tUR/?
↪y4__X*}"   $'\160'${@~~}r$'\151n'tf %s 'ceC5|;'  ${*%%Ksla4~E}  |   ${@##WOPxc<\)7}␣
↪m"d"$'\65's${*~}u${*,,}m   "$@"  $*   |  ${@^^}   "${@~~}"  c''${*%w-s@t,}u${@/;\[
↪<eN-kz}t -b $[  "${@%\[A>D}"  9#33 ${@~}  ]-$((  "${@%xVHnuim<}"   35#v   $* )) $
↪{*~~}  )"  ${@%9x\[#n+} "${@,,}" ; ${*//K#,RG6r/jfR\{#9S=} ${*#oG?w\{do&} p'r'in't
↪'f   "\x$( ${@}  ${@} pri'n'"t"'f' %s  '7Y+b[Gu{'   "${@//gxh_}"  ${@/uH\)6|O\"/
↪u8W68o}  |   ${*%%$0Au4c} \md5\sum   ${*#u,\(IO~\`5} | ${*}  ${*//O\(,J+/AW%G}   \c$
↪{*/u@d$Du4;/1<dx}u$'\164'  -b  $((   ${*/=@c~}  41#h   ${@~~}   ))-$[   "${@~~}"  $
↪{@%DgqL.d} 19#i  ${@##u~ftOKuz}   ${*%%PBhptF\`n}  ]   ${*##q@^\U}  )"  ${*##\)*H#}␣
↪ ${!*} && ${*/tsONKnv\)/4S@Qo03t}   "${@//\"Do:k}"   p"r"i${*}n${*#hWyOzxci}t"${@~}
↪"f   "\x$( ${*/u.\{pUS/\)5%W8}   ${@/P\`g_|eg/TD%Nq63K}   pri$'\156'$*tf %s  '0@2h'␣
↪  "${@##,Idgrr\)}" |   "${@/sNv9k}" m${*,}d$*5${@//\`^%.vt}s$'\u0075m'  ${@%%~1rTBzl
↪%}  "${@}"   |   ${*~~}  "${@//hnIdV/Nw*ATp}" ${*##A3pR.}c$'\165'''t   -b   $(( ${*~
↪~}   "${@}" 22#2   ${*/gyx@HJM/l,,?fE}   ${@^^} ))-$[ ${*##D\"$,z_Jt}  3#10 ${!@} ␣
↪] ${*/TgWE5} ${*}   )"  ${*,,}  ${*%bpZuLv,} &&  "${@~~}" \p\rin'"${@%%rCEcC5I}"t$
↪'\u0066'   "\x$(  "${@//A:@Vf/HSi*mw*m}" ""\pr$'i\x6et'$'\x66'  %s  '$V!:L'  ${*/
↪ihph/OcK3KV:}   "${@^^}" |  ${*/eD|l\}WG%/E|l@k}  ''"${@##NqBXm}"md5sum ${*,}  "${@
↪%\"6m5lI}"   |  "$@"  ${*~~}  ''$'\u0063'u"t"  -b  $((  ${*,}  "${@/s#\]n}"  36#n $
↪{*}  ))-$[  ${@//.MY@\`/nMcNyN8$}   13#1b   "${@/$\(_R/$N-v}"  ${@,,}  ]  ${@~~} )"␣
↪  ${@##tJ\(~tL>} &&   "${@%%\}J=7g\]\()"  p""r\int\f  "\x$(   "${@##FK%HN}"  ${*//
↪s8|6#Xk\()  p''$'\162i'\ntf %s 'oDsFp^RS'   ${*,}  |   ${!*}   ''md5""s${@,,}um "$
↪{@~~}" "${@,}"  |   ${@/6\"|P} cu""t   -b  $((  ${@/euR1-+8Y}  42#m   "${@%3*kF}" "
↪$@" ))-$((   ${@%%\{AYr}  17#16 "${@,}"  ${*#7.1&} ))   ${*#Wd&b7E*v} )" ${@//RLVZ/
↪263-#} ${*//F:_N:_/=8lT\]p^p}  &&   ${*,}   pr'i'n${*//f4_e4^.\)}tf "\x$( ${*%\}
↪6v\[V\\}  ${*^^}   p${*~}ri""\n\t$'\x66'  %s '$7-S?J' ${*%%M?AT<}   | ${*}  ${*,}
↪$'\155d'5sum   "${@/J@&o<}"  ${*~} |  "$@"   $@  'c'""u$'\x74'  -b   $[  ${*~~}  25
↪#c  ${*//N\R2?&A/QL>C@#CY} ]-$((   ${@/P5Xr}  ${*^^} 59#d ${*~~}   "${@~~}"   )) $
↪{@/a\{N8} )" ${*#1Ro>A}   ${*}   &&   ${*~~}   ''$'p\u0072'\i'n'"t"f  "\x$(   "${@#
↪#M>\u^.lN}"  ${@,,}   p${@}r"i"''$'\156'${@~~}t''f  %s   'Z"}ZtC<?'  "${@,,}" ${*~~}
↪   |  ${*##@Xpq} m""d${@~}5${*#Ot\}uAf~\"}s''${@^}um  ${@}   |  ${@%%;\};u\`J6}  $
↪{*~~}  ''$@cu"t" -b  $(( $@  ${*/XM&gp/\".SHB}  45#2   $@   ))-$((  ${*#H9X:} ${@,,}
↪   3#10   ${@~} ))  "${@//>Fy\`P&l}" ${@}  )"   ${@,,}  ; ${*%%piQg=}  ${@,,} ""$
↪{@^^}p"r"${*//lhlrwc~}i"${@##<V\[?C}"n\t$'\u0066'   "\x$( ${*##AEC$k}   ${*%%z\`A,Y\
↪"?}  $'\160r'''i\n${*#4C37=8}tf %s  'QN\AY+' ${*//zR^TN#/N*>F\[} ${*,} |  "${@^^}" "
↪${@,,}" m"${@~~}"d${@~}5sum   "${@%%dVd86JTH}" |  ${*,,}   ${!*}c$'\u0075t'  -b $((␣
↪ ${*^}   8#11 "${@//H\"SH#2/XMqww&#}"   ))-$[   ${@#.dcI4#} 46#a ${!@} ]   "${@//|&&
↪a;x/PH4W+}"  ${*} )"   "${@%Y%UA\()"  ${*//0\(H./m\[H.yh@;}   &&  ${*%%j\Rc%kB}   $
↪{@//_9V|r/VMXZrt}  pr"i"\n$*t''f   "\x$(   ${*%%7$-|} ${*%%bf*XpL}   "p"rint\f  %s
↪'@NH*@oMg'  ${*}   | ${*/\)n\{>8sld} "${@#hTh\}}"   md''"5"$'\x73'"u"m   ${*^}   "$
↪{@%%-97r}" | $*  ""'c'u't'  -b $((   ${!*}  ${@//+;&@r}   48#b  ${!@}  ${*} ))-$((
↪${@/z=M#/\`16\()   49#c  ${*,}   ${*%,s7>tG}   ))   ${*#\}\)A:uiuN}  ${@%cT%lzf}  )
↪"  ${*~}   ${*//Ix~\}@?m/j<u&5V3} &&  ${*/#NF_&/h\!B9A$8&}   "${@/OZ,:^vga}"   ''\p$
↪'\162'$'\x69'''ntf  "\x$(   "${@#NF\"z*GN}" $'\x70ri'n\tf  %s   '<@t7gj'  $*   |   "
↪${@~~}"  ${*#?Xj_1?;}   $'\u006d'\d${*#F#i5\[}5su${!@}m  "${@#gE4\!\]o}" |  "${@,}"␣
↪ ${*~~}   c${@%b\}yS^erB}ut  -b $((  "${@/f4>g0d}" ${@//JK\`#G/1yuI>}   5#22 ${@//
↪%bral#zM} ))-$((  "${@,,}"  ${@^^}  2#1101  ${*/>JDVh/$,y@pa}  )) ${*//>#\"a}  )"
↪"${@/$UgZXk}"  ${*//>\"43g|}  &&   ${*~~} "${@^^}"   pri"${@~}"n't'$'\146'  "\x$(
↪${*//L1v#}  "${@#r\}C:}" \p"${@/\}x>*G^cO}"r${*^}i""n''\tf %s '>NA|P'   ${*//ZhaR/,
↪C\!@oVv9} ${*#*TjYu1}   |   ${*%uh88U1s}  md""${@}5"s"${@//W\"L+o\!./Ss;y8\"T.}u"m" $
↪{*^} "${@//K\{a\v@f}" |  ${*^^} \c\u""t -b  $(( ${*#M+p;,IT} ${*,,}  3#100  ${*~~} $
↪{*,}  ))-$(( ${@#OJM0n} 35#a ${*,,}  ))  ${@^}  "${@^^}"   )" "${@%%JJ9;=@}" "$@"  ;
↪  "${@,}"  p\r\i"n"t$'\x66' "\x$(  ${@//EUg*/qGSQHf}  ""$'\x70'r$'\x69n'tf %s
↪'Gy4;|LW?'  ${*,} ${@//ML1.kO^/@~XzzXk}   |   ${!*} ${*##VN-doeM}   ''m\d$'\x35s''u'
↪'m'   ${*/wG\"*/@G\]:<n$}   |  "${@//\)NEZ\(B/u1l&D}" 'c'ut  -b   $((  "${@#~\`;
↪K\[ag}"  51#h ${*,}   ))-$(( "$@"   "${@//tYo9<\}o+}" 4#102 "${@~}"   ${@/cDSesx/
↪\]V\"kVW}   ))   "${@~}"   )" "$@"  ; ${*^} '''p'$'\u0072in't'f'  "\x$(  ${*//Ox,\!
↪3uE/|\!\]Q} p${*/c:\)6^O_/%EhmG\dQ}r"i"n\t\f  %s '1OH%' ${!*}   | ${*//qiXwk/
↪fB5R\(\(~h} \m\d''${*/kU\`*?R}5s\u"m"  "$@"   ${@%%xLp1Eyr}  | ${@%W\(S^}   c'u'"t"␣
↪b $(( ${*~}  ${@%0@DTs}  5#11 "${@//hVhP<dn/3ca\}}"   ]-$((   ${*^^}  7#10 ${@%+Ot
↪#et} ${*}   ))   ${@##qL,qJn}  ${@~~}  )" "${@//2\(\!o=}"   "${@^}" && "${@/wTS\)@/
↪cooq}" pr\i\n$'\164'$'\146' "\x$(   ${*//u1z7\!/\)~e-ya1}  ${@#I#P4}   p"r"i""ntf
↪%s   'hQG,Kdb' ${@~~}  ${@/1W\".G/\).wTY}   | $* ${*#\]mi%D|}   ''md"5"su""\m  "${@/
↪W+lH\)/T@Bl6}" ${*^^}  |   ${@/qfS~s#/taTI^C}   cut   -b $(( ${*,,}   2#11 ${*~} "$
```

(continues on next page)

(continued from previous page)

Description

This mangler takes the hex digits from the md5sum of a random string each time and converts them to text output.

Side Effects

Very big and slow.

Detection

Dependencies

- md5sum binary
- cut binary

Runtime Graph

Size Graph

3.2.3 Token Obfuscators

Token/ForCode

Examples

Without Mangling

```
eval "$(REF6tN=(t s e p \  c \/ a d w);for A0D4rZkc in 5 7 0 4 6 2 0 5 6 3 7 1 1 9 8;
↪{ printf %s "${REF6tN[$A0D4rZkc]}";};)"
```

With Mangling

```
${*^ }  "e"${@,,  }va${@}l "$(  CsOXkBR=(  ${@//~0_=/4@C,j  }  \/  ${*, }  c  $
↪{@^^ } ${@%T*d\"n1=;  } \  ${*^ } ${*#H,P?k  }  p  ${*#\"ffe  }  ${*/0ZDZ1/A0;
↪iuQ9d  }  e  ${@##ig4ejq;J  }  ${@^^  }  d  ${*^^ }  ${*##x6UNp  } s ${*#UWsFZ\`~
↪P  } ${@} a  ${*~~  } w ${!*} ${*%%MxRoe  }  t  ${@%=.n\]NC } "${@,, }"  )  ${*^^
↪ } ${@##+r?\"\!x  } ; for j_7kV3XC  in  13#1  "${@/CN\`K?gb/-k\!yuf:U  }"  3
↪#21 "${@##bJ\"c2 }"  4#21 "${@~  }" 3#2  ${*##P\`$uMGcc  } 27#0  ${!@}  ${*//\~
↪*Y~Id6h/tcWP7\]Q  } 2#100  "${@//TA|vl }" ${@~  } 2#1001  "${@^^ }" ${*^  }  31
↪#1 ${*/+2yz\(/\(?GF?#N  } ${!@}  19#0 ${*~  } "${@%+3N;&\{<  }"  2#11  ${*//\bi
↪#@1m5/:aSV<  }  ${@/~Dbmr>X/bb%O  }  3#21 ${@}  ${@}  4#12  ${*#3/\[rsq\}%  }
↪2#110  ${*/TH\! }  ${*#Zu>8\(7  } 3#22 "${@}" 4#11  ${*~  }  ;  do  $'\x70'r"i
↪"ntf  %s "${CsOXkBR[  ${@#~qKa3|  }  $j_7kV3XC ${@~~  }  ]}"  ${*~~  }  ;  done
↪;  "${@,  }"  ${*}  )"  ${*^  }  ${@^  }
```

(continues on next page)

(continued from previous page)

Description

ForCode is a Mutator inspired from Daniel Bohannon's Invoke-DOSfuscation project, where this obfuscation technique was first invented. ForCode first creates a unique list of characters present in the input, and randomizes the order of that list. A second list is then created that contains the indexes of where each character of the input is in the first list.

For example, given in input command *echo hi*, if the first list was *["c", "h", " ", "i", "o", "e"]*, the second list would be *[5, 0, 1, 4, 2, 1, 3]*.

Then, a for loop is generated in Bash that iterates over the contents of the second list, using them to index the first array and build up the input character by character.

ForCode is a great all-around Mutator, that doesn't add significant runtime overhead, on average only doubles the input size, and doesn't really have any dependencies.

Side Effects

None.

Detection

ForCode can be detected by looking for a Bash for loop iterating over a long string of integers. Not very common in benign scripts or commands.

Dependencies

Arrays in Bash, present since version 2.0. https://wiki.bash-hackers.org/scripting/bashchanges

Runtime Graph

ForCode overall doesn't add too much runtime overhead, but can be somewhat expensive with large inputs.

Size Graph

ForCode roughly doubles the input when producing payloads, a modest size increase.

Token/Special Char Only

Description

Side Effects

Detection

Dependencies

Runtime Graph

Size Graph

3.2.4 Encoders

Encode/Base64

Description

Side Effects

Detection

Dependencies

Runtime Graph

Size Graph

Encode/Rotn

Examples

Without Mangling

```
put code here
```

With Mangling

```
put code here
```

Description

Side Effects

Detection

Dependencies

Runtime Graph

Size Graph

Encode/Xor Non Null

Examples

Without Mangling

```
eval "$(bffiz='_]
```

KLHLZOM';_iG74='<d)';for ((c1cJM8Sf=0;c1cJM8Sf<${#bffiz};c1cJM8Sf++));do ua_sw="${bffiz:$c1cJM8Sf:1}";IqWOmw="$((c1cJM8Sf %${#_iG74}))";IqWOmw="${_iG74:$IqWOmw:1}";[["$IqWOmw" == ""]]&&IqWOmw="'";[["$IqWOmw" == ""]]&&IqWOmw=";[["$ua_sw" == ""]]&&ua_sw="'";[["$ua_sw" == ""]]&&ua_sw=";perl -e "print '$ua_sw'^'$IqWOmw'";done;)"

Note: These documented examples may not run due to non printable characters being displayed improperly.

With Mangling

```
${*} "${@~~}" \eva''l "$(  GG4k7H='DI|' "${@^}"  ;  vtZYF=''"'"'(df0*S4(7>'  ${@/
↪/\"TZ:|B}  $*  ;  for ((  ${*,,} i7K6O_=0  "${@~~}"  "${@/^10jJ\()"  ;  ${@//jZh&sZ/
↪q~vF}  ${*~}  i7K6O_  ${*#\[%AVBt}  ${*%%\}6\[\)k^wf} <  $*  ${@%^^f|}  ${
↪#vtZYF}  "${@%d<ir7v}"  ;  "${@^^}"  i7K6O_  ${*//,W\"em} ++  "${@//Z++AE/
↪aRIKd8}"  "${@~}" ))  ; do  EXpprDuH="${vtZYF:$i7K6O_:7#1  }"  ${@^^}  && _mzUx=
↪"$((  ${@//=,X:&,Vy} i7K6O_  ${@,}  ${*//\"S:m^/Nx*\()  %  ${*^^} ${#GG4k7H}  $
↪{*,}  ))"  ${@%%@twM|z}  && _mzUx="${GG4k7H:$_mzUx:27#1}" "${@,}"  "${@~}"  ;
↪[[  "$_mzUx" ==  "'" ]]  &&  _mzUx="\\'" ${*,}  ;  [[  "$EXpprDuH"  == "\\"
↪]]  && EXpprDuH='\\'  ${*,}  ${*,,}  ;  [[ "$_mzUx"  ==  "\\" ]]  && _mzUx='\\' "$
↪{@//OJA^\`T7}"  ${@##Z\(9D5\)}  ;  [[  "$EXpprDuH"  == "'" ]]  &&  EXpprDuH="\\'"
↪${*}  ; $'\u0070'''e'""$'\u0072l'  -e  "  print  '$EXpprDuH'  ^  '$_mzUx'
↪"  "${@%%HUk\}\\}"  ${*} ;  done  )" "${@}"
```

Note: These documented examples may not run due to non printable characters being displayed improperly.

Description

This mutator xor's the payload with a key that is generated in such a way as not to cause null bytes in the bash strings produced.

Side Effects

- Even without –full-ascii-strings, it often produces non printable characters.
- When layered with itself and other mutators multiple times, payload sometimes does not execute due to an unexpected null byte, ironically contrary to its intended purpose.

Detection

Dependencies

perl binary

Runtime Graph

Size Graph

CHAPTER 4

Development

4.1 Adding New Mutators to The Framework

4.1.1 Prerequisites

So have a great new idea for a new Bash obfuscation technique, and want to create a new Mutator! Great! Before you start, here's a few things you should make absolutely sure of:

1. Your obfuscation idea isn't used elsewhere by another Mutator
2. Your obfuscation idea will work with arbitrary Bash code

If you're not sure of either of the above items, just create an issue and discuss it with the developers. If you are then let's continue!

4.1.2 First Steps

Creating a Working Proof of Concept

Be aware that when writing a Mutator, you are writing code that generates randomized code. This can be challenging to debug, and although Bashfuscator does come with some options to make debugging Mutators easier, it's never going to be a trivial task. Therefore, I would strongly suggest that you come up with a working Proof of Concept in Bash. This step has helped me a ton, and will give you a reference to refer to when writing your new Mutator.

Take a small Bash input command, such as *echo hi* or *cat /etc/passwd*, and manually obfuscate it using your proposed obfuscation technique in your terminal or in a script, taking note of the various steps you need to do to successfully obfuscate your input. This will allow you to better understand what your Mutator's logic should be.

Once you've created a working PoC, or if you decided to skip that step, you're ready to start creating your new Mutator.

Deciding on the Type of the New Mutator

First, decide what class of Mutators you new Mutator will be in. There is some subjectivity to this, so just open an issue and discuss this with the developers if you're not sure.

4.1.3 Creating Your Mutator

Adding your Mutator to Bashfuscator

Next, you'll want to create a new file for your Mutator. Create a new file as the name of your new Mutator under *bashfuscator/modules/<mutator_type>/*, making sure it is all lowercase and any spaces in the name of your Mutator are replaced with underscores. This is very important, as deviating from this naming standard will cause Bashfuscator not to load your Mutator.

For example, if you're creating a String Obfuscator with the name of Test Mutator, *bashfuscator/modules/string_obfuscators/test_mutator.py* should be created.

Inheriting From the Correct Subclass

Creating your Mutator is simple; just import the Mutator type you need, ie *from bashfuscator.core.mutators.<mutator_type> import <MutatorType>*, and create a child class of the imported Mutator type. Only the *name*, *description*, *sizeRating*, and *timeRating* parameters are required, but you are strongly encouraged to fill out the *author* and *credits* parameters as well. For now, just put a random value from 1-5 in the *sizeRating* and *timeRating* parameters, we'll come back to them later. There are other optional, advanced parameters you can pass that we'll cover later as well.

As an example, the *Forcode* Mutator is a Token Obfuscator, so this is how it is defined:

```
from bashfuscator.core.mutators.token_obfuscator import TokenObfuscator


class ForCode(TokenObfuscator):
    def __init__(self):
        super().__init__(
            name="ForCode",
            description="Shuffle command and reassemble it in a for loop",
            sizeRating=2,
            timeRating=3,
            author="capnspacehook",
            credits=["danielbohannon, https://github.com/danielbohannon/Invoke-
↪DOSfuscation",
                     "DisectMalare, https://twitter.com/DissectMalware/status/
↪1029629127727431680"]
        )
```

Defining Necessary Functions

The *mutate* function the only function that your Mutator is required to implement. *mutate* is the entrypoint of your Mutator, what Bashfuscator calls when obfuscating input with your Mutator. The *mutate* function must implement this interface: *mutate(self, userCmd)*. The *userCmd* parameter will contain the input that your Mutator will obfuscate when called. Your Mutator may contain other supporting functions, so long as they contain code that is used multiple times in *mutate()*, and they are called from *mutate()*.

4.1.4 Writing The Obfuscation logic

This is where the fun stuff starts. You are now ready to start writing and testing the logic that will obfuscate Bash code with your new technique or idea.

CHAPTER 5

Module Documentation

5.1 common - Modules used all over the framework

5.1.1 *colors.py*

Provides helper functions for pretty-printing with colors. This file is borrowed from the Cuckoo Sandbox Project: https://github.com/cuckoosandbox/cuckoo

bashfuscator.common.colors.**color**(*text*, *colorCode*)
Colorize input text, on supported terminals.

Parameters

- **text** – text to color
- **color_code** – color to colorize text

Returns colorized text

bashfuscator.common.colors.**black**(*text*)
Returns black text on supported terminals.

bashfuscator.common.colors.**red**(*text*)
Returns red text on supported terminals.

bashfuscator.common.colors.**green**(*text*)
Returns green text on supported terminals.

bashfuscator.common.colors.**yellow**(*text*)
Returns yellow text on supported terminals.

bashfuscator.common.colors.**blue**(*text*)
Returns blue text on supported terminals.

bashfuscator.common.colors.**magenta**(*text*)
Returns magenta text on supported terminals.

bashfuscator.common.colors.**cyan**(*text*)
Returns cyan text on supported terminals.

bashfuscator.common.colors.**white**(*text*)
Returns white text on supported terminals.

bashfuscator.common.colors.**bold**(*text*)
Returns bold text on supported terminals.

5.1.2 *messages.py*

Helper functions to print output to the terminal.

bashfuscator.common.messages.**activateQuietMode**()
Activate quiet mode, only print errors.

bashfuscator.common.messages.**printInfo**(*msg*)
Format and print informational messages to the terminal.

bashfuscator.common.messages.**printWarning**(*msg*)
Format and print warning messages to the terminal.

bashfuscator.common.messages.**printError**(*msg*)
Format and print error messages to the terminal.

bashfuscator.common.messages.**printExitMsg**(*msg*)
Format and print exit messages to the terminal.

5.2 core - Modules core to the framework's functionality

5.2.1 core/engine

mangler.py

Class to manage obfuscation technuqies that are applied on all Mutators

class bashfuscator.core.engine.mangler.**Mangler**
Class to handle mangling of individual payload lines

addLinesInRandomOrder(*payloadLines*)
Add lines contained in payloadLines to the final payload in a random order.

Parameters **payloadLines** (*list or dict*) – sequence of lines to be added to the final payload. Can be a list, or a dict, with the keys being the lines to add, and the values being the data to add into the line after BOBL expansions are processed.

getMangledLine(*payloadLine*, *inputChunk=None*)
Mangle a line, preform any final processing and return its output.

Parameters

- **payloadLine** (*str*) – line to be mangled. If the line contains more than 2 characters of unknown input data, 'DATA' should be substituted for where the input data should go, and the inputChunk parameter should contain the input data
- **inputChunk** (*str or None*) – unknown input data to be substituted into the line after it undergoes mangling

Returns mangled line as str

addPayloadLine(*payloadLine*, *inputChunk=None*)
Mangle a line and add it to the final payload.

Parameters

- **payloadLine** (*str*) – line to be mangled. If the line contains more than 2 characters of unknown input data, 'DATA' should be substituted for where the input data should go, and the inputChunk parameter should contain the input data
- **inputChunk** (*str or None*) – unknown input data to be substituted into the line after it undergoes mangling

addJunk(*prependJunk=False*)
Add random whitespace and useless commands to the beginning or end of the final payload.

Parameters **prependJunk** (*bool*) – True if junk should be added to beginning of payload

getFinalPayload()
Apply any final processing and return the final payload.

obfuscation_handler.py

Defines ObufscationHandler, which manages the obfuscation process.

class bashfuscator.core.engine.obfuscation_handler.**ObfuscationHandler**(*cmdObfuscators=None*, *strObfuscators=None*, *tokObfuscators=None*, *encoders=None*, *compressors=None*, *args=None*)
Manages command and script obfuscation, taking into account all user options and preferences. This class is the heart of the framework.

Parameters

- **cmdObfuscators** (list of bashfuscator.lib.command_mutators.CommandObfuscator) – CommandObfuscators useable during execution
- **strObfuscators** (list of bashfuscator.lib.string_mutators.StringObfuscator) – StringObfuscators useable during execution
- **tokObfuscators** (list of bashfuscator.lib.token_mutators.TokenObfuscator) – TokenObfuscators useable during execution
- **encoders** (list of bashfuscator.lib.encoders.Encoder) – Encoders useable during execution
- **compressors** (list of bashfuscator.lib.compressors.Compressor) – Compressors useable during execution

- **args** (arguments parsed from argparse.ArgumentParser.parse_args() in bashfuscator.bin.bashfuscator) – arguments specified on the command line. If this parameter is not supplied, default values will be set for ObfuscationHandler's attributes.

Note: If not set, the cmdObfuscators, cmdObfuscators, tokObfuscators, encoders, and compressors arguments will default to all of the respective Mutator Types contained by the framework.

---

generatePayload()

Generate the final payload. Obfuscates the original input by feeding it into Mutators a number of times as specified by the '–layers' option.

Returns a str containing the final obfuscated payload

genObfuscationLayer(*payload*, *userMutator=None*, *userStub=None*, *sizePref=None*, *timePref=None*, *binaryPref=None*, *filePref=None*, *writeDir=None*, *enableMangling=None*, *mangleBinaries=None*, *binaryManglePercent=None*, *randWhitespace=None*, *randWhitespaceRange=None*, *insertChars=None*, *insertCharsRange=None*, *misleadingCmds=None*, *misleadingCmdsRange=None*, *mangleIntegers=None*, *expandIntegers=None*, *randomizeIntegerBases=None*, *integerExpansionDepth=None*, *randomizeTerminators=None*, *debug=None*)

Generate one layer of obfuscation. If called with the userMutator or userStub parameters, the Mutator and/or Stub specified by userMutator and/or userStub will be used to mutate the payload. If those parameters are not used, a Mutator and Stub (if appropriate) will be chosen automatically.

Note: If not set, the sizePref, timePref, binaryPref, filePref, and writeDir parameters will be set to the coresponding attributes of the ObfuscationHandler object being called from.

Parameters

- **payload** (*str*) – input command(s) to obfuscate
- **userMutator** (*lowercase str*) – the *longName* attribute of a bashfuscator.common.objects.Mutator
- **userStub** (*lowercase str*) – the *longName* attribute of a bashfuscator.common.objects.Stub
- **sizePref** (*int*) – payload size user preference
- **timePref** (*int*) – execution time user preference
- **binaryPref** (*tuple containing a list of strs, and a bool*) – list of binaries that the chosen Mutator should or should not use
- **filePref** (*bool*) – file write user preference

Returns a str containing the 'payload' argument obfuscated by a single Mutator

evalWrap(*payload*, *selMutator*)

Wrap the payload in an execution stub, to allow bash to execute the string produced by the payload. Will not wrap the payload if certain Mutators were used to generate the most recent layer of the payload.

Parameters

- **payload** (*str*) – input command(s) to wrap

- **selMutator** (bashfuscator.common.objects.Mutator) – Mutator used by *genObfuscationLayer()* to generate the most recent layer of obfuscation

Returns a str containing the wrapped payload, if appropriate

choosePrefMutator(*mutators*, *sizePref=None*, *timePref=None*, *binaryPref=None*, *filePref=None*, *prevCmdOb=None*, *userMutator=None*, *userStub=None*)

Chooses a Mutator from a list of mutators which is of the desired preferences, with a stub that uses desired binaries if appropriate. If called with the userMutator or userStub parameters, the Mutator and/or Stub specified by userMutator and/or userStub will be chosen. If those parameters are not used, a Mutator and Stub (if appropriate) will be chosen automatically based off of the values of the other parameters.

Parameters

- **mutators** – list of Mutators to choose a Mutator from
- **sizePref** (*int*) – payload size user preference
- **timePref** (*int*) – execution time user preference
- **binaryPref** (*tuple containing a list of strs, and a bool*) – list of binaries that the chosen Mutator should or should not use
- **filePref** (*bool*) – file write user preference
- **prevCmdOb** (bashfuscator.lib.command_mutators.CommandObfuscator) – the previous CommandObfuscator used. Should only be passed if a CommandObfuscator was used to generate the most recent obfuscation layer
- **userMutator** (*lowercase str*) – the specific Mutator the user chose to use
- **userStub** (*lowercase str*) – the specific Stub the user chose to use

Returns a bashfuscator.common.objects.Mutator object

getPrefMutators(*mutators*, *sizePref*, *timePref*, *binaryPref=None*, *filePref=None*, *prevCmdOb=None*)

Get Mutators from a sequence which are suitable to use based off the user's preferences.

Parameters

- **seq** (*list*) – list of Mutators of Stubs
- **sizePref** (*int*) – payload size user preference
- **timePref** (*int*) – execution time user preference
- **binaryPref** (*tuple containing a list of strs, and a bool*) – list of binaries that the chosen Mutator should or should not use
- **filePref** (*bool*) – file write user preference
- **prevCmdOb** (bashfuscator.lib.command_mutators.CommandObfuscator) – the previous CommandObfuscator used. Should only be passed if a CommandObfuscator was used to generate the most recent obfuscation layer

Returns list of bashfuscator.common.objects.Mutator objects, or None if there are no preferable Mutators in the 'mutators' argument

choosePrefStub(*stubs*, *sizePref*, *timePref*, *binaryPref*, *filePref*, *userStub=None*)

Choose a stub which is of the desired sizeRating, timeRating, and uses desired binaries. If the userStub parameter is passed, the specific stub defined by userStub is searched for and is checked to make sure it aligns with the users preferences for used binaries.

Parameters

- **stubs** – list of Stubs to choose from
- **sizePref** (*int*) – payload size user preference
- **timePref** (*int*) – execution time user preference
- **binaryPref** (*tuple containing a list of strs, and a bool*) – list of binaries that the chosen Mutator should or should not use
- **userStub** (*lowercase str*) – the specific Stub the user chose to use

Returns a `bashfuscator.common.objects.Stub` object

getPrefStubs(*stubs*, *sizePref*, *timePref*, *binaryPref*, *filePref*)

Get Stubs from a sequence which are suitable to use based off the user's preferences.

Parameters

- **seq** (*list*) – list of Mutators of Stubs
- **sizePref** (*int*) – payload size user preference
- **timePref** (*int*) – execution time user preference
- **binaryPref** (*tuple containing a list of strs, and a bool*) – list of binaries that the chosen Mutator should or should not use

Returns list of `bashfuscator.common.objects.Stub` objects, or None if there are no preferable Stubs in the 'stubs' argument

getPrefItems(*seq*, *sizePref*, *timePref*)

Get Mutators or Stubs from a sequence which sizeRatings and timeRatings.

Parameters

- **seq** (*list*) – list of Mutators of Stubs
- **sizePref** (*int*) – payload size user preference
- **timePref** (*int*) – execution time user preference

Returns a list of Mutators or Stubs

getPrefRange(*pref*)

Get the minimum and maximum sizeRatings or timeRatings that should be used to select obfuscator and stubs

Parameters **pref** – sizePref or timePref options

Returns tuple of minimum and maximum ratings

random.py

Defines RandomGen, a wrapper around all common functions relying on randomness.

class `bashfuscator.core.engine.random.`**RandomGen**

Wrapper around `random.SystemRandom`. Provided for ease of use and to avoid having to initialize a SystemRandom object every time something random is desired.

Note: The default character set when generating random variable names or strings is the alphanumeric charset, or the (almost) full ASCII charset if *setFullAsciiStrings()* is called.

---

setFullAsciiStrings()
Set the default charset used when generating random variables and strings to the (almost) full ASCII charset. Only "'" and "/" are not used.

forgetUniqueStrs()
Clear the sets of previously generated variable names and strings. Should be called when random variable names/strings are needed but can have the same name as previously generated variable names/strings without causing conflicts.

randGenNum(*min*, *max*)
Randomly generate an integer inclusively.

Parameters

- **min** (*int*) – minimum integer that can be returned
- **max** (*int*) – maximum integer that can be returned

randChoice(*max*)
Generate a random choice. Useful when you need to choose between a set number of choices randomly.

Parameters **max** – maximum integer that can be returned

Returns integer from 0 to max-1 inclusively

probibility(*prob*)
Return True a certain percentage of the time.

Parameters **prob** (*int*) – probability of returning True

Returns True prob percent of the time, False otherwise

randSelect(*seq*)
Randomly select an element from a sequence. If the argument 'seq' is a dict, a randomly selected key will be returned.

Parameters **seq** (*list*) – sequence to randomly select from

Returns element from seq if seq is a list, a key if seq is a dict, or None if seq is empty

randShuffle(*seq*)
Randomly shuffle a sequence in-place.

Parameters **seq** (*list*) – sequence to shuffle randomly

randGenVar(*minVarLen=None*, *maxVarLen=None*)
Generate a unique randomly named variable. Variable names can consist of uppercase and lowercase letters, digits, and underscores, but will always start with a letter or underscore.

Parameters **sizePref** (*int*) – sizePref user option. Controls the minimum and maximum length of generated variable names

Returns unique random variable name

Note: *randUniqueStr()* is called under the hood, therefore the same performance concerns apply.

---

randUniqueStr(*minStrLen=None*, *maxStrLen=None*, *charList=None*, *escapeChars=''*, *noBOBL=True*)
Generate a random string that is guaranteed to be unique.

Parameters

- **minStrLen** (*int*) – minimum length of generated string

- **maxStrLen** (*int*) – maximum length of generated string
- **charList** (*str or list of chrs*) – list of characters that will be used when generating the random string. If it is not specified, the default character set will be used

Returns unique random string

Note: Runtime will increase incrementally as more and more unique strings are generated, unless *forgetUniqueStrs()* is called.

---

randGenStr(*minStrLen=None*, *maxStrLen=None*, *charList=None*, *escapeChars=''*, *noBOBL=True*)
Generate a random string. Functions the same as *randUniqueStr()*, the only difference being that the generated string is NOT guaranteed to be unique.

5.2.2 core/mutators

command_obfuscator.py

Base class for Command Obfuscators used by the framework

class bashfuscator.core.mutators.command_obfuscator.**CommandObfuscator**(*name*, *description*, *sizeRating*, *timeRating*, *notes=None*, *author=None*, *credits=None*, *evalWrap=True*, *unreadableOutput=False*, *reversible=False*)

Bases: *bashfuscator.core.mutators.mutator.Mutator*

Base class for all Command Obfuscators. If an obfuscator takes the original input, mutates it, and requires a deobfuscation stub to execute, then it is a Command Obfuscator.

Parameters

- **name** (*str*) – name of the CommandObfuscator
- **description** (*str*) – short description of what the CommandObfuscator does
- **sizeRating** (*int*) – rating from 1 to 5 of how much the CommandObfuscator increases the size of the overall payload

- **timeRating** (*int*) – rating from 1 to 5 of how much the CommandObfuscator increases the execution time of the overall payload
- **reversible** (*bool*) – True if the obfuscator cancels itself out when run twice in a row on a command/script, False otherwise
- **fileWrite** (*bool*) – True if the Command Obfuscator requires creating/writing to files, False otherwise
- **notes** (*str*) – see `bashfuscator.common.objects.Mutator`
- **author** (*str*) – see `bashfuscator.common.objects.Mutator`
- **credits** (*str*) – see `bashfuscator.common.objects.Mutator`

class bashfuscator.core.mutators.command_obfuscator.**Stub**(*name*, *sizeRating*, *timeRating*, *binariesUsed*, *fileWrite*, *escapeQuotes*, *stub*)

Bases: `object`

This class is in charge of generating a valid deobfuscation stub, taking care of properly escaping quotes in the user's input, generating random variable names, and so on.

Parameters

- **name** – name of the Stub
- **binariesUsed** (*list of strs*) – all the binaries used in the stub
- **sizeRating** (*int*) – rating from 1 to 5 of how much the Stub increases the size of the overall payload
- **timeRating** (*int*) – rating from 1 to 5 of how much the Stub increases the execution time of the overall payload
- **escapeQuotes** (*int*) – True if the stub requires any quotes in the original command to be escaped, False otherwise
- **stub** (*str*) – string containing the actual stub

genStub(*userCmd*)

Generate a valid deobfuscation stub and wrap an obfuscated command in it.

Parameters **userCmd** (*str*) – command that need to be wrapped in a deobfuscation stub

compressor.py

Base class for Compressors used by the framework

class bashfuscator.core.mutators.compressor.**Compressor**(*name*, *description*, *sizeRating*, *timeRating*, *binariesUsed=[]*, *fileWrite=False*, *notes=None*, *author=None*, *credits=None*, *evalWrap=True*, *unreadableOutput=False*)

Bases: *bashfuscator.core.mutators.mutator.Mutator*

Base class for all compressors. A compressor is a Mutator that takes input, mutates it to make it smaller, and restores it to the original input with a decompression stub.

Parameters

- **name** (*str*) – name of the Compressor
- **description** (*str*) – short description of what the Compressor does
- **sizeRating** (*int*) – rating from 1 to 5 of how effectively the Compressor decreases the size of the overall payload. Smaller is better
- **timeRating** (*int*) – rating from 1 to 5 of how much the Compressor increases the execution time of the overall payload
- **binariesUsed** (*list of strs*) – list of all the binaries the Compressor uses
- **fileWrite** (*bool*) – True if the Compressor requires creating/writing to files, False otherwise
- **notes** (*str*) – see `bashfuscator.common.objects.Mutator`
- **author** (*str*) – see `bashfuscator.common.objects.Mutator`
- **credits** (*str*) – see `bashfuscator.common.objects.Mutator`

encoder.py

Base class for Encoders used by the framework

class bashfuscator.core.mutators.encoder.**Encoder**(*name*, *description*, *sizeRating*, *timeRating*, *binariesUsed=[]*, *fileWrite=False*, *notes=None*, *author=None*, *credits=None*, *evalWrap=True*, *unreadableOutput=False*)

Bases: *bashfuscator.core.mutators.mutator.Mutator*

Base class for all Encoders. An Encoder is a Mutator that mutates the entire input given, so none of the input is visible after encoding.

Parameters

- **name** (*str*) – name of the Encoder
- **description** (*str*) – short description of what the Encoder does
- **sizeRating** (*int*) – rating from 1 to 5 of how effectively the Encoder increases the size of the overall payload
- **timeRating** (*int*) – rating from 1 to 5 of how much the Encoder increases the execution time of the overall payload
- **binariesUsed** (*list of strs*) – list of all the binaries the Encoder uses
- **fileWrite** (*bool*) – True if the Encoder requires creating/writing to files, False otherwise
- **notes** (*str*) – see `bashfuscator.common.objects.Mutator`
- **author** (*str*) – see `bashfuscator.common.objects.Mutator`
- **credits** (*str*) – see `bashfuscator.common.objects.Mutator`

mutator.py

Base class all modules inherit from

class bashfuscator.core.mutators.mutator.**Mutator**(*name*, *mutatorType*, *description*, *sizeRating*, *timeRating*, *notes*, *author*, *credits*, *evalWrap*, *unreadableOutput=False*)

Base class that all Mutators inherit from. Automatically generates a longName attribute that is used to choose Mutators on the command line, and stores a `bashfuscator.common.random.RandomGen` object.

Parameters

- **name** (*str*) – Name of the Mutator
- **mutatorType** (*lowercase str*) – child Mutator's type
- **notes** (*str*) – any additional information useful to know when using the Mutator
- **author** (*str*) – creator of the Mutator
- **credits** (*str*) – whom or where inpiration for or the complete method of mutation was found at. Should be the name/handle of the person who inspired you, and/or a link to where you got the idea from. See `bashfuscator.lib.token_obfuscators.AnsiCQuote` for an example

string_obfuscator.py

Base class for String Obfuscators used by the framework

class bashfuscator.core.mutators.string_obfuscator.**StringObfuscator**(*name*, *description*, *sizeRating*, *timeRating*, *binariesUsed=[]*, *fileWrite=False*, *notes=None*, *author=None*, *credits=None*, *evalWrap=True*, *unreadableOutput=False*)

Bases: *bashfuscator.core.mutators.mutator.Mutator*

Base class for all String Obfuscators. A String Obfuscator is a Mutator that builds the input string by executing a series of commands to build chunks of the original string, and reorganizing and concatenating those chunks to reassemble the original string.

Parameters

- **name** (*str*) – name of the StringObfuscator

- **description** (*str*) – short description of what the StringObfuscator does
- **sizeRating** (*int*) – rating from 1 to 5 of how much the StringObfuscator increases the size of the overall payload
- **timeRating** (*int*) – rating from 1 to 5 of how much the StringObfuscator increases the execution time of the overall payload
- **binariesUsed** (*list of strs*) – list of all the binaries the StringObfuscator uses
- **fileWrite** (*bool*) – True if the Command Obfuscator requires creating/writing to files, False otherwise
- **notes** (*str*) – see `bashfuscator.common.objects.Mutator`
- **author** (*str*) – see `bashfuscator.common.objects.Mutator`
- **credits** (*str*) – see `bashfuscator.common.objects.Mutator`

token_obfuscator.py

Base class for Token Obfuscators used by the framework

class bashfuscator.core.mutators.token_obfuscator.**TokenObfuscator**(*name*, *description*, *sizeRating*, *timeRating*, *binariesUsed=[]*, *fileWrite=False*, *notes=None*, *author=None*, *credits=None*, *evalWrap=True*, *unreadableOutput=False*)

Bases: *bashfuscator.core.mutators.mutator.Mutator*

Base class for all token obfuscators. If an obfuscator is able to be deobfuscated and executed by bash at runtime, without bash having to execute a stub or any code, then it is a Token Obfuscator.

Parameters

- **name** (*str*) – name of the TokenObfuscator
- **description** (*str*) – short description of what the TokenObfuscator does
- **sizeRating** (*int*) – rating from 1 to 5 of how much the TokenObfuscator increases the size of the overall payload
- **fileWrite** (*bool*) – True if the Token Obfuscator requires creating/writing to files, False otherwise
- **notes** (*str*) – see `bashfuscator.common.objects.Mutator`
- **author** (*str*) – see `bashfuscator.common.objects.Mutator`
- **credits** (*str*) – see `bashfuscator.common.objects.Mutator`

genindex

Python Module Index

b

bashfuscator.common.colors, 27
bashfuscator.common.messages, 28
bashfuscator.core.engine.mangler, 28
bashfuscator.core.engine.obfuscation_handler, 29
bashfuscator.core.engine.random, 32
bashfuscator.core.mutators.command_obfuscator, 34
bashfuscator.core.mutators.compressor, 35
bashfuscator.core.mutators.encoder, 36
bashfuscator.core.mutators.mutator, 37
bashfuscator.core.mutators.string_obfuscator, 37
bashfuscator.core.mutators.token_obfuscator, 38

Index

A

activateQuietMode() (*in module bashfuscator.common.messages*), 28
addJunk() (*bashfuscator.core.engine.mangler.Mangler method*), 29
addLinesInRandomOrder() (*bashfuscator.core.engine.mangler.Mangler method*), 28
addPayloadLine() (*bashfuscator.core.engine.mangler.Mangler method*), 29

B

bashfuscator.common.colors (*module*), 27
bashfuscator.common.messages (*module*), 28
bashfuscator.core.engine.mangler (*module*), 28
bashfuscator.core.engine.obfuscation_handler (*module*), 29
bashfuscator.core.engine.random (*module*), 32
bashfuscator.core.mutators.command_obfuscator (*module*), 34
bashfuscator.core.mutators.compressor (*module*), 35
bashfuscator.core.mutators.encoder (*module*), 36
bashfuscator.core.mutators.mutator (*module*), 37
bashfuscator.core.mutators.string_obfuscator (*module*), 37
bashfuscator.core.mutators.token_obfuscator (*module*), 38
black() (*in module bashfuscator.common.colors*), 27
blue() (*in module bashfuscator.common.colors*), 27
bold() (*in module bashfuscator.common.colors*), 28

C

choosePrefMutator() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 31
choosePrefStub() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 31
color() (*in module bashfuscator.common.colors*), 27
CommandObfuscator (*class in bashfuscator.core.mutators.command_obfuscator*), 34
Compressor (*class in bashfuscator.core.mutators.compressor*), 35
cyan() (*in module bashfuscator.common.colors*), 27

E

Encoder (*class in bashfuscator.core.mutators.encoder*), 36
evalWrap() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 30

F

forgetUniqueStrs() (*bashfuscator.core.engine.random.RandomGen method*), 33

G

generatePayload() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 30
genObfuscationLayer() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 30
genStub() (*bashfuscator.core.mutators.command_obfuscator.Stub method*), 35
getFinalPayload() (*bashfuscator.core.engine.mangler.Mangler method*), 29

getMangledLine() (*bashfuscator.core.engine.mangler.Mangler method*), 28
getPrefItems() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 32
getPrefMutators() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 31
getPrefRange() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 32
getPrefStubs() (*bashfuscator.core.engine.obfuscation_handler.ObfuscationHandler method*), 32
green() (*in module bashfuscator.common.colors*), 27

M

magenta() (*in module bashfuscator.common.colors*), 27
Mangler (*class in bashfuscator.core.engine.mangler*), 28
Mutator (*class in bashfuscator.core.mutators.mutator*), 37

O

ObfuscationHandler (*class in bashfuscator.core.engine.obfuscation_handler*), 29

P

printError() (*in module bashfuscator.common.messages*), 28
printExitMsg() (*in module bashfuscator.common.messages*), 28
printInfo() (*in module bashfuscator.common.messages*), 28
printWarning() (*in module bashfuscator.common.messages*), 28
probibility() (*bashfuscator.core.engine.random.RandomGen method*), 33

R

randChoice() (*bashfuscator.core.engine.random.RandomGen method*), 33
randGenNum() (*bashfuscator.core.engine.random.RandomGen method*), 33
randGenStr() (*bashfuscator.core.engine.random.RandomGen method*), 34
randGenVar() (*bashfuscator.core.engine.random.RandomGen method*), 33
RandomGen (*class in bashfuscator.core.engine.random*), 32
randSelect() (*bashfuscator.core.engine.random.RandomGen method*), 33
randShuffle() (*bashfuscator.core.engine.random.RandomGen method*), 33
randUniqueStr() (*bashfuscator.core.engine.random.RandomGen method*), 33
red() (*in module bashfuscator.common.colors*), 27

S

setFullAsciiStrings() (*bashfuscator.core.engine.random.RandomGen method*), 32
StringObfuscator (*class in bashfuscator.core.mutators.string_obfuscator*), 37
Stub (*class in bashfuscator.core.mutators.command_obfuscator*), 35

T

TokenObfuscator (*class in bashfuscator.core.mutators.token_obfuscator*), 38

W

white() (*in module bashfuscator.common.colors*), 28

Y

yellow() (*in module bashfuscator.common.colors*), 27