Obfuscator

Tom Rochette <tom.rochette@coreteks.org>

July 24, 2025 — b5600af2

0.1 Context

0.2 Learned in this study

0.3 Things to explore

1 Overview

- Encapsulate the application within a .phar
- Use obfuscation (can only obfuscate variables inside function/methods as anything else must remain with the same name in order to allow for dynamically called methods (or simply do not support such code))

2 Requirements

- Must not make it easy to extract the original source code
- Must checksum itself for modification
- Should prevent usage by sharing a single license (TBD?)
- Should not be possible to reuse a trial forever in a VM

3 Issues

- Relying on opcode (to act as a binary) probably implies you are dependent on the php version used. Furthermore, it also means that the extension must be available and enabled for the code to work
- Given the current implementation of op*cache*, the cache cannot be reused on other machines as it contains the system ID as part of an opcache file header

4 PHP OPCache

4.1 Header

- OPCACHE
- System ID
- Timestamp
- Checksum

-> zend_accel_load_script (persistent_script, from_memory)

5 Obfuscation

- Replace classes methods with (generated) traits that contain one or many of the classes method
- Compression/optimization by creating variables

5.1 Issues

- Even with obfuscation, PSR0/4 based code is too easy to distinguish (no reason to obfuscate libraries)

5.2 Ideas

- Find some way to transform the original source such that it is in symbiose with a client identifier (and thus cannot be removed)
  - See one-way function
- Obfuscate php built-in functions
- Zend Host ID limited deployment: It appears that Zend Guard allows the software to be limited to run on only specific "Zend Host" (what are those?)
  - Zend Guard Loader is a custom extension? for PHP which requires a licence_path to work

6 Packaging

- Replace variables within functions with obfuscated names
- Concatenate all files into a single file (or bundle of files)
  - Pre-process require/include, or do not allow their use (as the files will not be available)
- Randomize file order
- Scramble method order
- Inject junk code
  - Insert junk code with encrypted user identification
- Generate a differently scrambled version per download/user

7 See also

8 References

- https://github.com/naneau/php-obfuscator
- https://www.pelock.com/products/autoit-obfuscator
- https://developers.google.com/closure/compiler/
- http://proguard.sourceforge.net/
- http://www.zend.com/en/products/zend-guard