A Glance at Interpreted Language Bytecode Trickery

By: Chris Lyne

Sr. Research Engineer

Hello world!

Chris Lyne
@lynerc
Sr. Research Engineer
0-Day Research

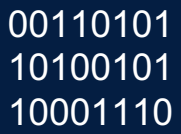
Background

- Druva inSync
  - Endpoint backup software
  - Python

- Nagios XI
  - Enterprise Server and Network Monitoring Software
  - PHP

Topics

Interpreted Languages

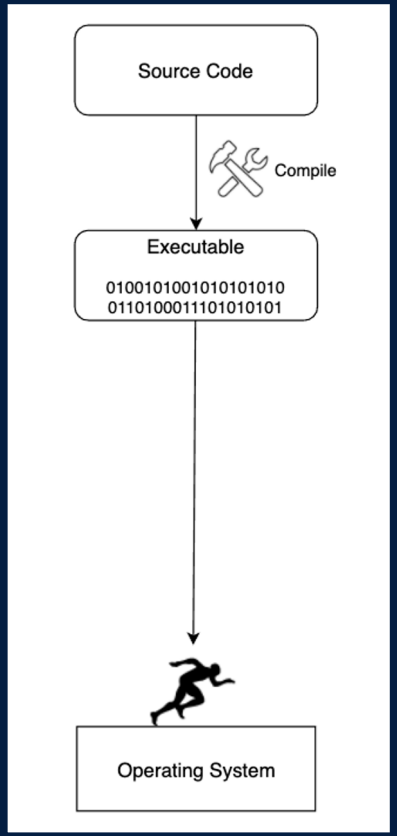


Bytecode

Protections Encountered

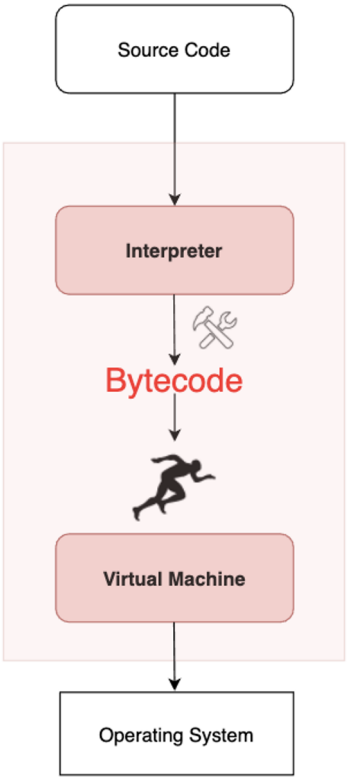
How I Bypassed Them

Compiled



C, C++, Go, ...

Interpreted



Python, PHP, Ruby, ...

Python Opcode Remapping

Druva inSync

The protection

Code Objects

Fixing the opcodes

Druva inSync is built with py2exe

Procmon Output

inSync.exe	11088	IRP_MJ_READ	C:\Program Files (x86)\Druva\inSync\python27.dll	SUCCESS
inSync.exe	11088	IRP_MJ_READ	C:\Program Files (x86)\Druva\inSync\library.zip	SUCCESS

library.zip

_psutil_windows.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_osx_support.pyc	2/11/2020 5:35 PM	Compiled Python File	12 KB
_ntfsutil.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_mysql_exceptions.pyc	2/11/2020 5:35 PM	Compiled Python File	4 KB
_mysql.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_multiprocessing.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_MozillaCookieJar.pyc	2/11/2020 5:35 PM	Compiled Python File	5 KB
_LWPCookieJar.pyc	2/11/2020 5:35 PM	Compiled Python File	6 KB
_inSyncDU.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_hashlib.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_elementtree.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_ctypes.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_cffi_backend.pyc	2/11/2020 5:35 PM	Compiled Python File	1 KB
_abcoll.pyc	2/11/2020 5:35 PM	Compiled Python File	24 KB
__future__.pyc	2/11/2020 5:35 PM	Compiled Python File	5 KB
zip3	4/15/2020 10:46 AM	File folder	
zip2	4/15/2020 10:46 AM	File folder	
xml	4/15/2020 10:48 AM	File folder	
winreg	4/15/2020 10:48 AM	File folder	
win32com	4/15/2020 10:48 AM	File folder	

What's a .pyc file?

- A byte-compiled Python file
- Helps to speed up the load time
- Can be decompiled back into source code (e.g. with uncompyle6)

```
C:\Users\lyner>uncompyle6 C:\Python27\Lib\struct.pyc
# uncompyle6 version 3.6.5
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (Intel)]
# Embedded file name: c:\python27\lib\struct.py
# Compiled at: 2017-02-13 21:38:06
from _struct import *
from _struct import _clearcache
from _struct import __doc__
# okay decompiling C:\Python27\Lib\struct.pyc
```

Python Opcode Remapping

It won’t decompile!

```
C:\Users\lyner>uncompyle6 C:\Users\lyner\Desktop\library\struct.pyc
Traceback (most recent call last):
  File "c:\python27\lib\runpy.py", line 174, in _run_module_as_main
    "__main__", fname, loader, pkg_name)
  File "c:\python27\lib\runpy.py", line 72, in _run_code
    exec code in run_globals
  File "C:\Python27\Scripts\uncompyle6.exe\__main__.py", line 9, in <module>
  File "c:\python27\lib\site-packages\uncompyle6\bin\uncompile.py", line 194, in main_bin
    **options)
  File "c:\python27\lib\site-packages\uncompyle6\main.py", line 327, in main
    do_fragments,
  File "c:\python27\lib\site-packages\uncompyle6\main.py", line 187, in decompile_file
    filename, code_objects
  File "c:\python27\lib\site-packages\xdis\load.py", line 116, in load_module
    get_code=get_code,
  File "c:\python27\lib\site-packages\xdis\load.py", line 152, in load_module_from_file_object
    % (ord(magic[0:1]) + 256 * ord(magic[1:2]), filename)
ImportError: Unknown magic number 62216 in C:\Users\lyner\Desktop\library\struct.pyc
```

.pyc file format

```
print 'Hello world!'
```

```
>>> import py_compile
>>> py_compile.compile('hello.py')
```

Magic String Timestamp Code Object

```
sh-3.2$ xxd hello.pyc
00000000: 03f3 0d0a 839a a85e 6300 0000 0000 0000  .......^c.......
00000010: 0001 0000 0040 0000 0073 0900 0000 6400  .....@...s....d.
00000020: 0047 4864 0100 5328 0200 0000 730c 0000  .GHd..S(....s...
00000030: 0048 656c 6c6f 2077 6f72 6c64 214e 2800  .Hello world!N(.
00000040: 0000 0028 0000 0000 2800 0000 0028 0000  ...(....(....(..
00000050: 0000 7308 0000 0068 656c 6c6f 2e70 7974  ..s....hello.pyt
00000060: 0800 0000 3c6d 6f64 756c 653e 0100 0000  ....<module>....
00000070: 7400 0000 00                             t....
```

Magic Number List

```
#     Python 2.5c2: 62131 (fix wrong code: for x, in ... in listcomp/genexp)
#     Python 2.6a0: 62151 (peephole optimizations and STORE_MAP opcode)
#     Python 2.6a1: 62161 (WITH_CLEANUP optimization)
#     Python 2.7a0: 62171 (optimize list comprehensions/change LIST_APPEND)
#     Python 2.7a0: 62181 (optimize conditional branches:
#                          introduce POP_JUMP_IF_FALSE and POP_JUMP_IF_TRUE)
#     Python 2.7a0  62191 (introduce SETUP_WITH)
#     Python 2.7a0  62201 (introduce BUILD_SET)
#     Python 2.7a0  62211 (introduce MAP_ADD and SET_ADD)
#     Python 3000:   3000
#                    3010 (removed UNARY_CONVERT)
#                    3020 (added BUILD_SET)
#                    3030 (added keyword-only parameters)
#                    3040 (added signature annotations)
#                    3050 (print becomes a function)
#                    3060 (PEP 3115 metaclass syntax)
#                    3061 (string literals become unicode)
#                    3071 (PEP 3109 raise changes)
#                    3081 (PEP 3137 make __file__ and __name__ unicode)
#                    3091 (kill str8 interning)
#                    3101 (merge from 2.6a0, see 62151)
#                    3103 (__file__ points to source file)
#     Python 3.0a4: 3111 (WITH_CLEANUP optimization).
```


Source: https://github.com/python/cpython/blob/5d1f32d33ba24d0aa87235ae40207bb57778388b/Lib/importlib/_bootstrap_external.py#L200

Opcodes

```
169  def_op('LOAD_FAST', 124)        # Local variable number
170  haslocal.append(124)
171  def_op('STORE_FAST', 125)       # Local variable number
172  haslocal.append(125)
173  def_op('DELETE_FAST', 126)      # Local variable number
174  haslocal.append(126)
175
176  def_op('RAISE_VARARGS', 130)    # Number of raise arguments (1, 2, or 3)
177  def_op('CALL_FUNCTION', 131)    # #args
178  def_op('MAKE_FUNCTION', 132)    # Flags
179  def_op('BUILD_SLICE', 133)      # Number of items
180
181  def_op('LOAD_CLOSURE', 135)
182  hasfree.append(135)
183  def_op('LOAD_DEREF', 136)
184  hasfree.append(136)
185  def_op('STORE_DEREF', 137)
```

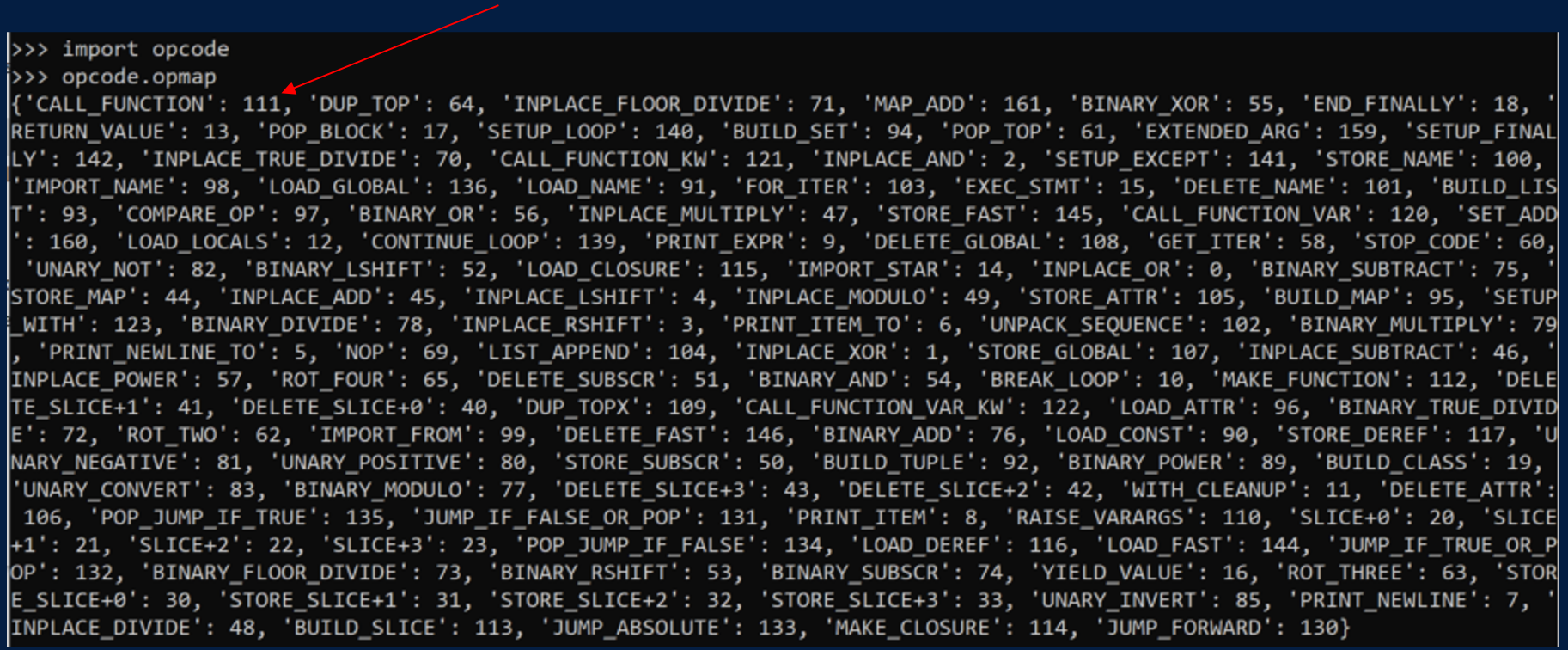
Source: https://github.com/python/cpython/blob/master/Lib/opcode.py#L177

Normal Python 2.7 Opcodes

```
C:\>python
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import opcode
>>> opcode.opmap
{'CALL_FUNCTION': 131, 'DUP_TOP': 4, 'INPLACE_FLOOR_DIVIDE': 28, 'MAP_ADD': 147, 'BINARY_XOR': 65, 'END_FINALLY': 88, 'R
ETURN_VALUE': 83, 'POP_BLOCK': 87, 'SETUP_LOOP': 120, 'BUILD_SET': 104, 'POP_TOP': 1, 'EXTENDED_ARG': 145, 'SETUP_FINALL
Y': 122, 'INPLACE_TRUE_DIVIDE': 29, 'CALL_FUNCTION_KW': 141, 'INPLACE_AND': 77, 'SETUP_EXCEPT': 121, 'STORE_NAME': 90, '
IMPORT_NAME': 108, 'LOAD_GLOBAL': 116, 'LOAD_NAME': 101, 'FOR_ITER': 93, 'EXEC_STMT': 85, 'DELETE_NAME': 91, 'BUILD_LIST
': 103, 'COMPARE_OP': 107, 'BINARY_OR': 66, 'INPLACE_MULTIPLY': 57, 'STORE_FAST': 125, 'CALL_FUNCTION_VAR': 140, 'SET_AD
D': 146, 'LOAD_LOCALS': 82, 'CONTINUE_LOOP': 119, 'PRINT_EXPR': 70, 'DELETE_GLOBAL': 98, 'GET_ITER': 68, 'STOP_CODE': 0,
 'UNARY_NOT': 12, 'BINARY_LSHIFT': 62, 'LOAD_CLOSURE': 135, 'IMPORT_STAR': 84, 'INPLACE_OR': 79, 'BINARY_SUBTRACT': 24,
'STORE_MAP': 54, 'INPLACE_ADD': 55, 'INPLACE_LSHIFT': 75, 'INPLACE_MODULO': 59, 'STORE_ATTR': 95, 'BUILD_MAP': 105, 'SET
UP_WITH': 143, 'BINARY_DIVIDE': 21, 'INPLACE_RSHIFT': 76, 'PRINT_ITEM_TO': 73, 'UNPACK_SEQUENCE': 92, 'BINARY_MULTIPLY':
 20, 'PRINT_NEWLINE_TO': 74, 'NOP': 9, 'LIST_APPEND': 94, 'INPLACE_XOR': 78, 'STORE_GLOBAL': 97, 'INPLACE_SUBTRACT': 56,
 'INPLACE_POWER': 67, 'ROT_FOUR': 5, 'DELETE_SUBSCR': 61, 'BINARY_AND': 64, 'BREAK_LOOP': 80, 'MAKE_FUNCTION': 132, 'DEL
ETE_SLICE+1': 51, 'DELETE_SLICE+0': 50, 'DUP_TOPX': 99, 'CALL_FUNCTION_VAR_KW': 142, 'LOAD_ATTR': 106, 'BINARY_TRUE_DIVI
DE': 27, 'ROT_TWO': 2, 'IMPORT_FROM': 109, 'DELETE_FAST': 126, 'BINARY_ADD': 23, 'LOAD_CONST': 100, 'STORE_DEREF': 137,
'UNARY_NEGATIVE': 11, 'UNARY_POSITIVE': 10, 'STORE_SUBSCR': 60, 'BUILD_TUPLE': 102, 'BINARY_POWER': 19, 'BUILD_CLASS': 8
9, 'UNARY_CONVERT': 13, 'BINARY_MODULO': 22, 'DELETE_SLICE+3': 53, 'DELETE_SLICE+2': 52, 'WITH_CLEANUP': 81, 'DELETE_ATT
R': 96, 'POP_JUMP_IF_TRUE': 115, 'JUMP_IF_FALSE_OR_POP': 111, 'PRINT_ITEM': 71, 'RAISE_VARARGS': 130, 'SLICE+0': 30, 'SL
ICE+1': 31, 'SLICE+2': 32, 'SLICE+3': 33, 'POP_JUMP_IF_FALSE': 114, 'LOAD_DEREF': 136, 'LOAD_FAST': 124, 'JUMP_IF_TRUE_O
R_POP': 112, 'BINARY_FLOOR_DIVIDE': 26, 'BINARY_RSHIFT': 63, 'BINARY_SUBSCR': 25, 'YIELD_VALUE': 86, 'ROT_THREE': 3, 'ST
ORE_SLICE+0': 40, 'STORE_SLICE+1': 41, 'STORE_SLICE+2': 42, 'STORE_SLICE+3': 43, 'UNARY_INVERT': 15, 'PRINT_NEWLINE': 72
, 'INPLACE_DIVIDE': 58, 'BUILD_SLICE': 133, 'JUMP_ABSOLUTE': 113, 'MAKE_CLOSURE': 134, 'JUMP_FORWARD': 110}
```

Source: https://github.com/python/cpython/blob/master/Lib/opcode.py#L177

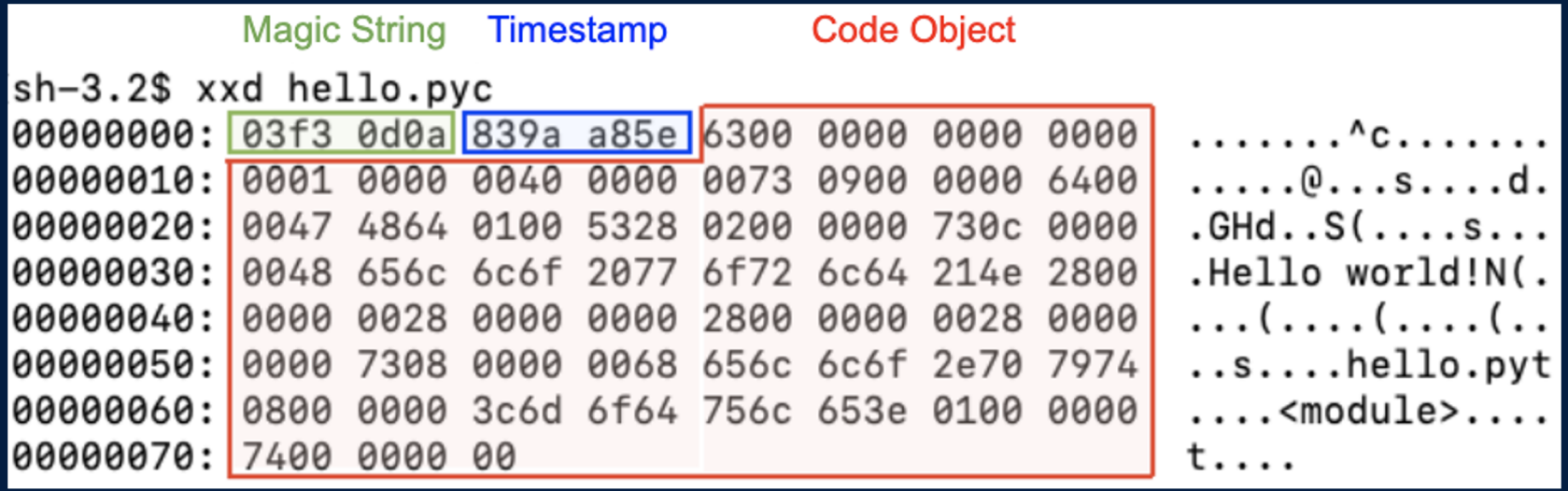
Druva Opcodes



Source: https://github.com/python/cpython/blob/master/Lib/opcode.py#L177

Python Opcode Remapping

.pyc format again. But code object



Code object is executable

```
>>> f = open('hello.pyc', 'rb')
>>> bytes = f.read()
>>> f.close()
```

```
>>> code_bytes = bytes[8:]
>>> import marshal
>>> code_object = marshal.loads(code_bytes)
>>> code_object
<code object <module> at 032F0BF0, file "hello.py", line 1>
>>> exec(code_object)
Hello world!
```

```
>>> code_object.co_code
'd\x00\x00GHd\x01\x00S'
```

Raw bytecode

Disassembling Code Object in hello.pyc

```
>>> import dis
>>> dis.dis(code_object)
  1           0 LOAD_CONST               0 ('Hello world!')
              3 PRINT_ITEM
              4 PRINT_NEWLINE
              5 LOAD_CONST               1 (None)
              8 RETURN_VALUE
```

```
>>> code_object.co_consts
('Hello world!', None)
```

Nested Code Object

```
class HelloClass(object):
    def __init__(self):
        pass
    def sayHello(self):
        print "Oh hai!"
```

```
>>> dis.dis(code_obj)
  1           0 LOAD_CONST               0 ('HelloClass')
              3 LOAD_NAME                0 (object)
              6 BUILD_TUPLE              1
              9 LOAD_CONST               1 (<code object HelloClass at 03B515C0, file "helloclass.py", line 1>)
             12 MAKE_FUNCTION            0
             15 CALL_FUNCTION            0
             18 BUILD_CLASS
             19 STORE_NAME               1 (HelloClass)
             22 LOAD_CONST               2 (None)
             25 RETURN_VALUE
```

More Nested Code Objects

```
>>> code_obj.co_consts
('HelloClass', <code object HelloClass at 03B515C0, file "helloclass.py", line 1>, None)
```

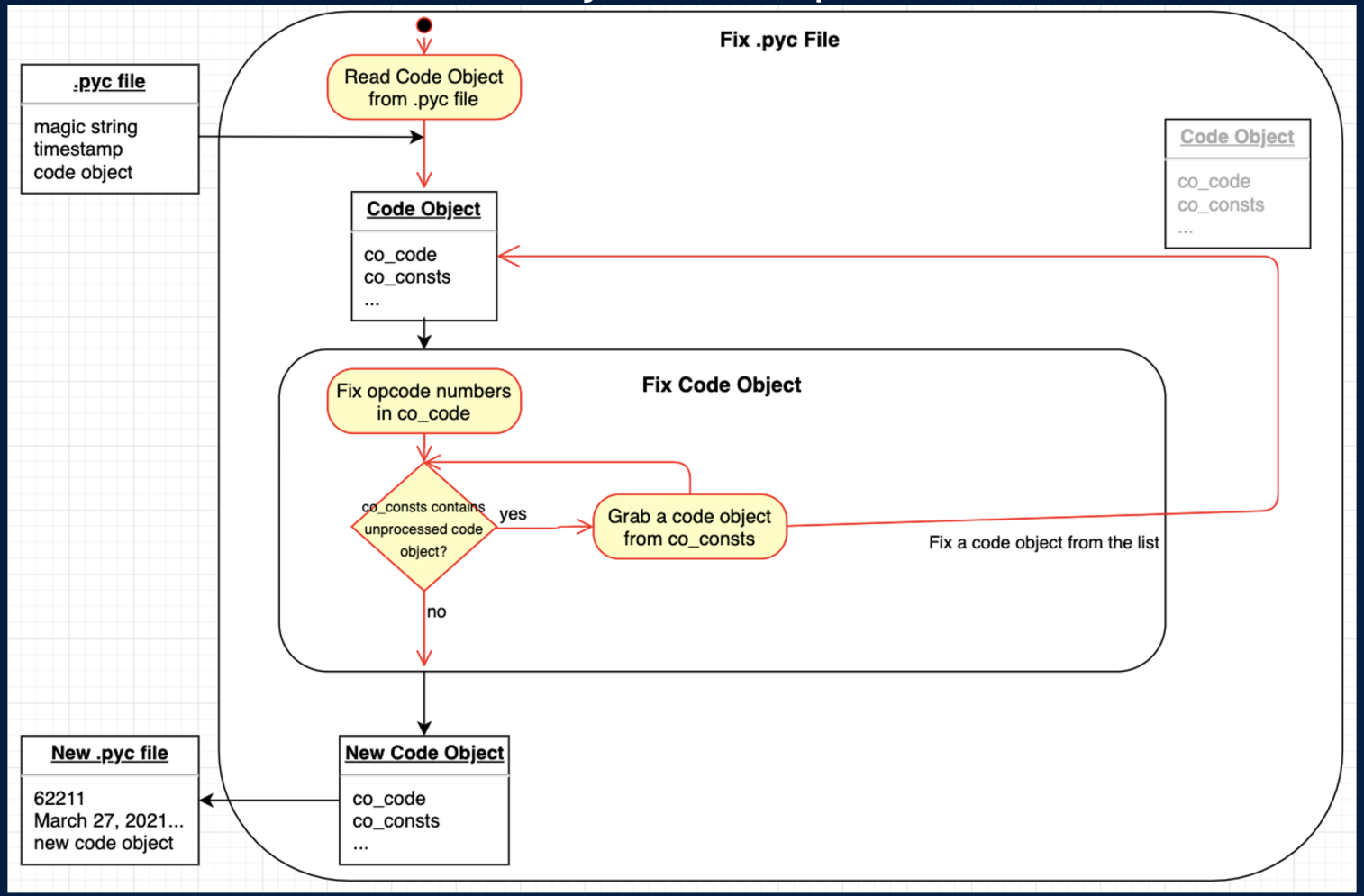
```
>>> dis.dis(code_obj.co_consts[1])
  1           0 LOAD_NAME                0 (__name__)
              3 STORE_NAME               1 (__module__)

  2           6 LOAD_CONST               0 (<code object __init__ at 03B516E0, file "helloclass.py", line 2>)
              9 MAKE_FUNCTION            0
             12 STORE_NAME               2 (__init__)

  4          15 LOAD_CONST               1 (<code object sayHello at 03B51608, file "helloclass.py", line 4>)
             18 MAKE_FUNCTION            0
             21 STORE_NAME               3 (sayHello)
             24 LOAD_LOCALS
             25 RETURN_VALUE
```

Python Opcode Remapping

Remap the Opcodes in Code Objects

Druva Opcode		Instruction Name		Normal Python 2.7 Opcode
111	<-->	CALL_FUNCTION	<-->	131
64	<-->	DUP_TOP	<-->	4
71	<-->	IN_PLACE_FLOOR_DIVIDE	<-->	28
161	<-->	MAP_ADD	<-->	147
...				

Recursively fix the opcodes...



PHP - SourceGuardian

The protection

PHP Internals

VLD

Solution

Do you read SourceGuardian?

```
<?php
if(!function_exists('sg_load')){$__v=phpversion();$__x=explode('.',$__v);$__v2=$__x[0].'.'.(int)$__x[1];$__u=strtolower(substr(php_uname(),0,3));$__ts=(@constant('PHP_ZTS') || @constant('ZEND
_THREAD_SAFE')?'ts':'');$__f=$__f0='ixed.'.$__v2.$__ts.'.'.$__u;$__ff=$__ff0='ixed.'.$__v2.'.'.(int)$__x[2].$__ts.'.'.$__u;$__ed=@ini_get('extension_dir');$__e=$__e0=@realpath($__ed);$__dl=fu
nction_exists('dl') && function_exists('file_exists') && @ini_get('enable_dl') && !@ini_get('safe_mode');if($__dl && $__e && version_compare($__v,'5.2.5','<') && function_exists('getcwd') &&
function_exists('dirname')){$__d=$__d0=getcwd();if(@$__d[1]==':') {$__d=str_replace('\\','/',substr($__d,2));$__e=str_replace('\\','/',substr($__e,2));}$__e.=($__h=str_repeat('/..',substr_cou
nt($__e,'/')));$__f='/ixed/'.$__f0;$__ff='/ixed/'.$__ff0;while(!file_exists($__e.$__d.$__ff) && !file_exists($__e.$__d.$__f) && strlen($__d)>1){$__d=dirname($__d);}if(file_exists($__e.$__d.$_
_ff)) dl($__h.$__d.$__ff); else if(file_exists($__e.$__d.$__f)) dl($__h.$__d.$__f);}if(!function_exists('sg_load') && $__dl && $__e0){if(file_exists($__e0.'/'.$__ff0)) dl($__ff0); else if(fil
e_exists($__e0.'/'.$__f0)) dl($__f0);}if(!function_exists('sg_load')){$__ixedurl='http://www.sourceguardian.com/loaders/download.php?php_v='.urlencode($__v).'&php_ts='.($__ts?'1':'0').'&php_i
s='.@constant('PHP_INT_SIZE').'&os_s='.urlencode(php_uname('s')).'&os_r='.urlencode(php_uname('r')).'&os_m='.urlencode(php_uname('m'));$__sapi=php_sapi_name();if(!$__e0) $__e0=$__ed;if(functi
on_exists('php_ini_loaded_file')) $__ini=php_ini_loaded_file(); else $__ini='php.ini';if((substr($__sapi,0,3)=='cgi')||($__sapi=='cli')||($__sapi=='embed')){$__msg="\nPHP script '".__FILE__."
' is protected by SourceGuardian and requires a SourceGuardian loader '".$__f0."' to be installed.\n\n1) Download the required loader '".$__f0."' from the SourceGuardian site: ".$__ixedurl."\
n2) Install the loader to ";if(isset($__d0)){$__msg.=$__d0.DIRECTORY_SEPARATOR.'ixed';}else{$__msg.=$__e0;if(!$__dl){$__msg.="\n3) Edit ".$__ini." and add 'extension=".$__f0."' directive";}}$
__msg.="\n\n";}else{$__msg="<html><body>PHP script '".__FILE__."' is protected by <a href=\"http://www.sourceguardian.com/\">SourceGuardian</a> and requires a SourceGuardian loader '".$__f0."
' to be installed.<br><br>1) <a href=\"".$__ixedurl."\" target=\"_blank\">Click here</a> to download the required '".$__f0."' loader from the SourceGuardian site<br>2) Install the loader to "
;if(isset($__d0)){$__msg.=$__d0.DIRECTORY_SEPARATOR.'ixed';}else{$__msg.=$__e0;if(!$__dl){$__msg.="<br>3) Edit ".$__ini." and add 'extension=".$__f0."' directive<br>4) Restart the web server"
;}}$__msg.="</body></html>";}die($__msg);exit();}}return sg_load('D0AE83944756C0F8AAQAAAAXAAAABGgAAACABAAAAAAAAAD/6AsLEZ+5lPPXqsaT1Ikbic0ttplQ5TchQpA5M8TsO+remlDjilIoRjIsvQ/p8LagX/OI8bEQH+PK+
ZULJKVsvuzDxmzYW3ozz9HlwAHjItDuRqUwuiYgC1ot4+Etafxja/fEX5w3Smo2AAAAEAMAAIH9Z+xbi+r44VZ3l88i1M54BDLx5MDSacE4L9+qlY/ZI5671SLIKIgXkP0Zv2vyiZFMyNEEUFn6z5HOVGWLmaPZeL7R7U75rOQpbzIzBAsBa+OvwbeefG9R
b9OUEDbHZzgFpv63tPsBCttNpG8pWHmUMJ5YB+PgiMBkJyayKmUWAAJIJIop4mNZHMOe7WtRimOdMA6s7cW+Jiq4ekP/dh+k5WWTKljgqEmsQU3b5ErAO2Age6FP8+ggP8+SIk3SJc4UdNMsNMf04+KC5Gmuzsnnpsqr6Oq651X0CHGH7uITV5GLail1uR
34Yd9p/aLZULS/xS7i5ELndHQD7//NOql+aiVbr4DcmInJZ7FkuAIFSApVifA7grdybob6+L7hfnoiJ+02LNNpp/qOTAPB4IbJEDnFZ8I/O5dI0SEWUVLPP1YqpFc471ixVW2GKZ2AgxHHo3HKS0rb3jWHcsTgM59cWXTuQdfqza8EnVElyZwXqhN6u1ABo
7sn3P/Rv15Vzow/7U6yMKUjg5RDi7FQooPyzBIBKtyF8VAa9CS0IM2bnmkK9uDjdf6Vhn5yMo2R6prUIdzsjJ1bfTzHuv5ZyHwcphxRcV3f66JXP6cREqW7dJq3taoZgKDCQheA17lJNsEMp1dPlPXbf1QJsDPlZjbkJ8B4VXEradzkJxFdcb8qC86DwNLd
FvEuO+45padvJqFw8LEkDiY7nIkRpF9ax9kaIgUCQrA+Tx5ijrfKp7xcMjftbOW+daxxkFxiqAN/fIKSQZtIEDGUhMPOCOhBudY9Ex/Ch1aIUurz86jZkTjqeUvhoCXyQ2MQeG6mqEZOfQhGm9fxFpUJpPu5CD+5Sg7SW+KyIKKpo2wbymNt6e5RqdeacZK
nmzHeG+eUHBC6+Al1t9Mt2YVpotqOkf9a5sOJLg4PJA5wWa83DKoLf05A/9hEAeCDUo+2O5E8A/FJdX7TFFxeN6EN+5irry/fMmWuQuRRui2ot5g/vSLIi0P3xjDsGIjVJpmH0XBqSFmwY9OwAnhOM/eHXv3HiDhgYkAAAAA');
```

sg_load()

```
return sg_load('52C4625F2E337708AAQAAAAXAAAABHAAAACABAAAAAAAAAD/NWy5LNQ/PM1LphDNvP8yRsZRX8ZY01OX
vN/pyGABx+Jn7GUCVSLLV5e9dzUAAADQEQAAggYdPCB3QaE1O32w3w5ijzs2GqpvwA/+GjFX3pkhD8a/p3x8qIe+kJ8bJb7Z
NSuWAyH+4XZ2h0QccREqHGLh0D/GvwAxaWG4PGUkm8FfFDsGRw/Q1mEn5wTuymOKEWfrKEdjVR0zUASa57piG/tvStI5xEPa
eyRiToD8zPNvlBbe+gC7VyG6RcaOX+olOU1Yn54HeI8/ms1M4ikPtHKkxdH+LNJu4wf95/yRXkm0x2K/vz9ypiLLgKZ3YwXH
ojgP6382DQOol6Okks6iRyjYcLvdB5Q6rx4AzBBMGubelcA0P9SCb4X+3gdgjJJko+JgXWvDgPulRO1VDNQ5JvwhllkO3bWA
b28BoZnbiLjYKX0GT/HP6mbR1u7v/Z2F5MbFPVxlJFvNdbEhX3f63YzCIFZjVWCEPw5GTgi7Sbfs7GrbohO0zn6cWnSo2zrn
836v0eXM9AeeyzoGAcXc9m/oxk8YYPxwcMUJZ3Ww9+ywiLqIqV8YOYPV94wuQ9Ig3k/xYhpRTPlMsKLVvTq0BrwFcvOmBrUq
```

“Our PHP encoder protects your PHP code by compiling the PHP source code into a binary bytecode format, which is then supplemented with an encryption layer.”

So sg_load() first *decrypts* and then *executes* the bytecode

PHP - SourceGuardian

PHP Bytecode

- PHP does not implement a concept of a .pyc file
- However, there are PHP extensions that precompile scripts
  - “OPcache improves PHP performance by storing precompiled script bytecode in shared memory, thereby removing the need for PHP to load and parse scripts on each request.”
- PHP source code is compiled into bytecode which is executed by the Zend VM runtime

Source: https://www.php.net/manual/en/intro.opcache.php

PHP Hooks and Extensions

- “PHP and the Zend Engine provide many different “hooks” for extensions that allow extension developers to control the PHP runtime in ways that are not available from PHP userland.”
  - Zend_compile_file()
    - PHP code transformed into bytecode
  - Zend_execute()
    - Bytecode is executed
  - Overwrite Opcode handlers
    - Functions to handle a specific operation (e.g. ECHO)
  - Also misc functions
    - .e.g var_dump()
- Useful for debuggers such as XDebug

Source: https://www.phpinternalsbook.com/php7/extensions_design/hooks.html

PHP - SourceGuardian

The protection PHP Internals VLD Solution

Vulcan Logic Dumper (VLD) Extension

```php
<?php
    echo "Hello world!\n";
?>
```

```
filename:       hello.php
function name:  (null)
number of ops:  2
compiled vars:  none
line     #* E I O op                           fetch          ext  return  operands
---------------------------------------------------------------------------------
   3     0*       ECHO                                                   OP1[  IS_CONST (80737094) 'Hello+world%21%0A' ]
   6     1*       RETURN                                                 OP1[  IS_CONST (80737095) 1 ]

Hello world!
```

https://github.com/derickr/vld

VLD - zend_compile_file() hook

```
op_array = old_compile_file (file_handle, type TSRMLS_CC);

if (VLD_G(path_dump_file)) {
        fprintf(VLD_G(path_dump_file), "subgraph cluster_file_%p { label=\"file %s\";\n", op_array, op_array->filename ? ZSTRING_VALUE(op_array->filename) : "__main");
}
if (op_array) {
        vld_dump_oparray (op_array TSRMLS_CC);
}
```

1. Compile source code from PHP file
2. vld_dump_oparray() - Dump the opcodes

VLD - zend_compile_file() hook - sg_load()

```
$ php -dvld.dump_paths=0 hello.php
filename:       /home/osboxes/sg_tests/hello.php
function name:  (null)
number of ops:  332
compiled vars:  !0 = $__v, !1 = $__x, !2 = $__v2, !3 = $__u, !4 = $__ts, !5 = $__f, !6 = $__f0, !7 = $__ff, !8 = $__ff0, !9 = $__ed, !10 = $__e, !11 = $__e0, !12 = $__dl, !13 = $__d, !14 = $__d0, !15 = $__h, !16 = $__ixedurl, !17 = $__sapi, !18 = $__ini, !19 = $__msg
line     #* E I O op                           fetch          ext  return  operands
-------------------------------------------------------------------------------------
   2     0*       SEND_VAL                                             OP1[  IS_CONST (83408084) 'sg_load' ]
         1*       DO_FCALL                                       1   RES[  IS_VAR $0 ]       OP1[  IS_CONST (83408085) 'function_exists' ]
         2*       BOOL_NOT                                           RES[  IS_TMP_VAR ~1 ]       OP1[  IS_VAR $0 ]
         3*       JMPZ                                                 OP1[  IS_TMP_VAR ~1 ] OP2[ , ->328 ]
         4*       DO_FCALL                                       0   RES[  IS_VAR $2 ]       OP1[  IS_CONST (83408087) 'phpversion' ]
         5*       ASSIGN                                             RES[  ]         OP1[  IS_CV !0 ] OP2[ ,  IS_VAR $2 ]
         6*       SEND_VAL                                             OP1[  IS_CONST (83408088) '.' ]
         7*       SEND_VAR                                             OP1[  IS_CV !0 ]
         8*       DO_FCALL                                       2   RES[  IS_VAR $4 ]       OP1[  IS_CONST (83408089) 'explode' ]
         9*       ASSIGN                                             RES[  ]         OP1[  IS_CV !1 ] OP2[ ,  IS_VAR $4 ]
        10*       FETCH_DIM_R                                        RES[  IS_VAR $6 ]       OP1[  IS_CV !1 ] OP2[ ,  IS_CONST (83408090) 0 ]
        11*       CONCAT                                             RES[  IS_TMP_VAR ~7 ]       OP1[  IS_VAR $6 ] OP2[ ,  IS_CONST (83408092) '.' ]
        12*       FETCH_DIM_R                                        RES[  IS_VAR $8 ]       OP1[  IS_CV !1 ] OP2[ ,  IS_CONST (83408093) 1 ]
        13*       CAST                                           1   RES[  IS_TMP_VAR ~9 ]       OP1[  IS_VAR $8 ] OP2[  IS_UNUSED  ]
        14*       CONCAT                                             RES[  IS_TMP_VAR ~10 ]       OP1[  IS_TMP_VAR ~7 ] OP2[ ,  IS_TMP_VAR ~9 ]
        15*       ASSIGN                                             RES[  ]         OP1[  IS_CV !2 ] OP2[ ,  IS_TMP_VAR ~10 ]
        16*       DO_FCALL                                       0   RES[  IS_VAR $12 ]       OP1[  IS_CONST (83408094) 'php_uname' ]
        17*       SEND_VAR_NO_REF                                6   RES[  IS_UNUSED  ]         OP1[  IS_VAR $12 ] OP2[  IS_UNUSED  ]
        18*       SEND_VAL                                             OP1[  IS_CONST (83408095) 0 ]
        19*       SEND_VAL                                             OP1[  IS_CONST (83408097) 3 ]
        20*       DO_FCALL                                       3   RES[  IS_VAR $13 ]       OP1[  IS_CONST (83408098) 'substr' ]
        21*       SEND_VAR_NO_REF                                6   RES[  IS_UNUSED  ]         OP1[  IS_VAR $13 ] OP2[  IS_UNUSED  ]
        22*       DO_FCALL                                       1   RES[  IS_VAR $14 ]       OP1[  IS_CONST (83408099) 'strtolower' ]
        23*       ASSIGN                                             RES[  ]         OP1[  IS_CV !3 ] OP2[ ,  IS_VAR $14 ]
        24*       BEGIN_SILENCE                                      RES[  IS_TMP_VAR ~16 ]       OP1[  IS_UNUSED  ] OP2[  IS_UNUSED  ]
        25*       SEND_VAL                                             OP1[  IS_CONST (83408100) 'PHP_ZTS' ]
        26*       DO_FCALL                                       1   RES[  IS_VAR $17 ]       OP1[  IS_CONST (83408102) 'constant' ]
        27*       END_SILENCE                                        RES[  IS_UNUSED  ]         OP1[  IS_TMP_VAR ~16 ] OP2[  IS_UNUSED  ]
        28*       JMPNZ_EX                                           RES[  IS_TMP_VAR ~18 ]       OP1[  IS_VAR $17 ] OP2[ , ->34 ]
        29*       BEGIN_SILENCE                                      RES[  IS_TMP_VAR ~19 ]       OP1[  IS_UNUSED  ] OP2[  IS_UNUSED  ]
        30*       SEND_VAL                                             OP1[  IS_CONST (83408103) 'ZEND_THREAD_SAFE' ]
        31*       DO_FCALL                                       1   RES[  IS_VAR $20 ]       OP1[  IS_CONST (83408104) 'constant' ]
        32*       END_SILENCE                                        RES[  IS_UNUSED  ]         OP1[  IS_TMP_VAR ~19 ] OP2[  IS_UNUSED  ]
        33*       BOOL                                               RES[  IS_TMP_VAR ~18 ]       OP1[  IS_VAR $20 ]
        34*       JMPZ                                                 OP1[  IS_TMP_VAR ~18 ] OP2[ , ->37 ]
        35*       QM_ASSIGN                                          RES[  IS_TMP_VAR ~21 ]       OP1[  IS_CONST (83408105) 'ts' ]
        36*       JMP                                                  OP1[ ->38 ]
        37*       QM_ASSIGN                                          RES[  IS_TMP_VAR ~21 ]       OP1[  IS_CONST (83408107) '' ]
        38*       ASSIGN                                             RES[  ]         OP1[  IS_CV !4 ] OP2[ ,  IS_TMP_VAR ~21 ]
        39*       CONCAT                                             RES[  IS_TMP_VAR ~23 ]       OP1[  IS_CONST (83408108) 'ixed.' ] OP2[ ,  IS_CV !2 ]
        40*       CONCAT                                             RES[  IS_TMP_VAR ~24 ]       OP1[  IS_TMP_VAR ~23 ] OP2[ ,  IS_CV !4 ]
        41*       CONCAT                                             RES[  IS_TMP_VAR ~25 ]       OP1[  IS_TMP_VAR ~24 ] OP2[ ,  IS_CONST (83408109) '.' ]
        42*       CONCAT                                             RES[  IS_TMP_VAR ~26 ]       OP1[  IS_TMP_VAR ~25 ] OP2[ ,  IS_CV !3 ]
        43*       ASSIGN                                             RES[  IS_VAR $27 ]       OP1[  IS_CV !6 ] OP2[ ,  IS_TMP_VAR ~26 ]
        44*       ASSIGN                                             RES[  ]         OP1[  IS_CV !5 ] OP2[ ,  IS_VAR $27 ]
        45*       CONCAT                                             RES[  IS_TMP_VAR ~28 ]       OP1[  IS_CONST (83408110) 'ixed.' ] OP2[ ,  IS_CV !2 ]
```

Visualizing the Hook

Executing a SourceGuardian Protected File

$ php file.php	zend_compile_file()	zend_execute()	sg_load()	zend_execute()
SourceGuardian-encoded file is launched by PHP interpreter. The file contains all of the code necessary to decode the compiled, encrypted bytecode. This includes a call to sg_load(<arg contains encrypted bytecode>);	file.php is **compiled** into bytecode (zend_op_array). This includes a call to sg_load();	Bytecode from previous step is executed. This, in turn, calls sg_load().	sg_load() fires. It decrypts the encrypted bytecode, and calls zend_execute() to execute the bytecode.	The decrypted bytecode is executed. Hook this! 

2nd zend_execute() hook - no opcodes!?

```
static void vld_execute(zend_op_array *op_array TSRMLS_DC)
#endif
{
        php_printf("\nexecute()\n");
        vld_dump_oparray (op_array TSRMLS_CC);
        old_execute(op_array);
}
```

```
execute()
filename:       hello.php
function name:  (null)
number of ops:  3
compiled vars:  none
line     #* E I O op                                   fetch          ext  return  operands
-------------------------------------------------------------------------------------------

Hello world!
```

But why? ... Time to debug

1

zend_execute()

```
Breakpoint 1, execute (op_array=0x7ffff5b7e918) at php-
src/Zend/zend_vm_execute.h:343
343 {
(gdb) c
Continuing.


execute()
filename:       hello.php
function name:  (null)
number of ops:  3
compiled vars:  none
line     #* E I O op                           fetch          ext
return  operands
-----------------------------------------------------------------------
```

2

zend_execute()

```
Breakpoint 1, execute (op_array=0x7ffff5b85340) at php-
src/Zend/zend_vm_execute.h:343
343 {
(gdb) p op_array
$1 = (zend_op_array *) 0x7ffff5b85340
(gdb) p *op_array
$2 = {type = 2 '\002', function_name = 0x0, scope = 0x0, fn_flags =
134217728, prototype = 0x0, num_args = 0, required_num_args = 0,
arg_info = 0x0, refcount = 0x7ffff5b805f8, opcodes = 0x7ffff5b7ea18,
last = 3, vars = 0x0, last_var = 0, T = 0, brk_cont_array = 0x0,
  last_brk_cont = 0, try_catch_array = 0x0, last_try_catch = 0,
static_variables = 0x0, this_var = 4294967295, filename =
0x7ffff5b7eab8 "hello.php", line_start = 0, line_end = 0,
doc_comment = 0x0, doc_comment_len = 0, early_binding = 4294967295,
  literals = 0x7ffff5b85440, last_literal = 2, run_time_cache = 0x0,
last_cache_slot = 0, reserved = {0x555555f4e450, 0x0, 0x0, 0x0}}
```

Line Numbers are all 0?!

```
(gdb) p op_array->opcodes[0]
$4 = {handler = 0x7ffff4a09280, op1 = {constant = 4122471032, var =
4122471032, num = 4122471032, hash = 140737315859064, opline_num =
4122471032, jmp_addr = 0x7ffff5b7ea78, zv = 0x7ffff5b7ea78, literal
= 0x7ffff5b7ea78, ptr = 0x7ffff5b7ea78}, op2 = {constant = 0,
    var = 0, num = 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv =
0x0, literal = 0x0, ptr = 0x0}, result = {constant = 0, var = 0, num
= 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal =
0x0, ptr = 0x0}, extended_value = 0, lineno = 0, opcode = 42 '*',
  op1_type = 0 '\000', op2_type = 0 '\000', result_type = 0 '\000'}


(gdb) p op_array->opcodes[1]
$5 = {handler = 0x5555558dfaa0 <ZEND_ECHO_SPEC_CONST_HANDLER>, op1 =
{constant = 4122498112, var = 4122498112, num = 4122498112, hash =
140737315886144, opline_num = 4122498112, jmp_addr = 0x7ffff5b85440,
zv = 0x7ffff5b85440, literal = 0x7ffff5b85440,
    ptr = 0x7ffff5b85440}, op2 = {constant = 0, var = 0, num = 0,
hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0,
ptr = 0x0}, result = {constant = 0, var = 0, num = 0, hash = 0,
opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0, ptr = 0x0},
  extended_value = 0, lineno = 0, opcode = 40 '(', op1_type = 1
'\001', op2_type = 0 '\000', result_type = 0 '\000'}


(gdb) p op_array->opcodes[2]
$6 = {handler = 0x5555558cd390 <ZEND_RETURN_SPEC_CONST_HANDLER>, op1
= {constant = 4122498152, var = 4122498152, num = 4122498152, hash =
140737315886184, opline_num = 4122498152, jmp_addr = 0x7ffff5b85468,
zv = 0x7ffff5b85468, literal = 0x7ffff5b85468,
    ptr = 0x7ffff5b85468}, op2 = {constant = 0, var = 0, num = 0,
hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0,
ptr = 0x0}, result = {constant = 0, var = 0, num = 0, hash = 0,
opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0, ptr = 0x0},
  extended_value = 0, lineno = 0, opcode = 62 '>', op1_type = 1
'\001', op2_type = 0 '\000', result_type = 0 '\000'}
```

```
struct _zend_op {
        opcode_handler_t handler;
        znode_op op1;
        znode_op op2;
        znode_op result;
        ulong extended_value;
        uint lineno;
        zend_uchar opcode;
        zend_uchar op1_type;
        zend_uchar op2_type;
        zend_uchar result_type;
};
```

/*

```
if (op.lineno == 0) {
        return;
}
```

*/

New output! ... New JMP?

Encoded File

```
filename:       hello.php
function name:  (null)
number of ops:  3
compiled vars:  none
line     #* E I O op                           fetch          ext  return  operands
-------------------------------------------------------------------------------
   0     0*        JMP                                                    OP1[ ->2 ]
         1*        ECHO                                                   OP1[  IS_CONST (33780247) 'Hello+world%21%0A' ]
         2*        RETURN                                                 OP1[  IS_CONST (33780248) 1 ]

Hello world!
```

Not Encoded (expected output)

```
filename:       hello.php
function name:  (null)
number of ops:  2
compiled vars:  none
line     #* E I O op                           fetch          ext  return  operands
-------------------------------------------------------------------------------
   3     0*        ECHO                                                   OP1[  IS_CONST (80737094) 'Hello+world%21%0A' ]
   6     1*        RETURN                                                 OP1[  IS_CONST (80737095) 1 ]
```

Another sample...

```php
<?php


$num = rand(0, 1);
if ($num == 1)
{
    echo "1\n";
}
else
{
    echo "0\n";
}


?>
```

Wait a sec. JMPZNZ?

Encoded

```
filename:       if_else.php
function name:  (null)
number of ops:  12
compiled vars:  !0 = $num
line     #* E I O op                           fetch          ext  return  operands
-------------------------------------------------------------------------------------
   0     0*        JMP                                                    OP1[ ->4 ]
         1*        ECHO                                                   OP1[  IS_CONST (89088013) '0%0A' ]
         2*        JMP                                                    OP1[ ->3 ]
         3*        SEND_VAL                                               OP1[  IS_CONST (89088006) 0 ]
         4*        SEND_VAL                                               OP1[  IS_CONST (89088008) 1 ]
         5*        DO_FCALL                                    2  RES[  IS_VAR $0 ]       OP1[  IS_CONST (89088009) 'rand' ]
         6*        ASSIGN                                         RES[  ]         OP1[  IS_CV !0 ] OP2[ ,  IS_VAR $0 ]
         7*        IS_EQUAL                                       RES[  IS_TMP_VAR ~2 ]       OP1[  IS_CV !0 ] OP2[ ,  IS_CONST (89088010) 1 ]
         8*        JMPZNZ                                     11          OP1[  IS_TMP_VAR ~2 ] OP2[ , ->6 ]
         9*        RETURN                                             OP1[  IS_CONST (89088014) 1 ]
        10*        ECHO                                               OP1[  IS_CONST (89088011) '1%0A' ]
        11*        JMP                                                OP1[ ->3 ]
```

Not Encoded (expected output)

```
filename:       if_else.php
function name:  (null)
number of ops:  10
compiled vars:  !0 = $num
line     #* E I O op                           fetch          ext  return  operands
-------------------------------------------------------------------------------------
   3     0  E >   SEND_VAL                                              OP1[  IS_CONST (49233527) 0 ]
         1        SEND_VAL                                              OP1[  IS_CONST (49233528) 1 ]
         2        DO_FCALL                                    2  RES[  IS_VAR $0 ]       OP1[  IS_CONST (49233529) 'rand' ]
         3        ASSIGN                                         RES[  ]         OP1[  IS_CV !0 ] OP2[ ,  IS_VAR $0 ]
   5     4        IS_EQUAL                                       RES[ IS_TMP_VAR ~2 ]       OP1[  IS_CV !0 ] OP2[ ,  IS_CONST (49233530) 1 ]
   6     5      > JMPZ                                                 OP1[  IS_TMP_VAR ~2 ] OP2[ , ->8 ]
   7     6    >   ECHO                                                 OP1[  IS_CONST (49233532) '1%0A' ]
   8     7      > JMP                                                  OP1[ ->9 ]
  11     8    >   ECHO                                                 OP1[  IS_CONST (49233533) '0%0A' ]
  15     9    > > RETURN                                               OP1[  IS_CONST (49233534) 1 ]
```

Hello World - Back to the Debugger - Op Handlers

```
(gdb) p op_array->opcodes[0]
$4 = {handler = 0x7ffff4a09280, op1 = {constant = 4122471032, var =
4122471032, num = 4122471032, hash = 140737315859064, opline_num =
4122471032, jmp_addr = 0x7ffff5b7ea78, zv = 0x7ffff5b7ea78, literal
= 0x7ffff5b7ea78, ptr = 0x7ffff5b7ea78}, op2 = {constant = 0,
    var = 0, num = 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv =
0x0, literal = 0x0, ptr = 0x0}, result = {constant = 0, var = 0, num
= 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal =
0x0, ptr = 0x0}, extended_value = 0, lineno = 0, opcode = 42 '*',
  op1_type = 0 '\000', op2_type = 0 '\000', result_type = 0 '\000'}


(gdb) p op_array->opcodes[1]
$5 = {handler = 0x5555558dfaa0 <ZEND_ECHO_SPEC_CONST_HANDLER>, op1 =
{constant = 4122498112, var = 4122498112, num = 4122498112, hash =
140737315886144, opline_num = 4122498112, jmp_addr = 0x7ffff5b85440,
zv = 0x7ffff5b85440, literal = 0x7ffff5b85440,
    ptr = 0x7ffff5b85440}, op2 = {constant = 0, var = 0, num = 0,
hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0,
ptr = 0x0}, result = {constant = 0, var = 0, num = 0, hash = 0,
opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0, ptr = 0x0},
  extended_value = 0, lineno = 0, opcode = 40 '(', op1_type = 1
'\001', op2_type = 0 '\000', result_type = 0 '\000'}


(gdb) p op_array->opcodes[2]
$6 = {handler = 0x5555558cd390 <ZEND_RETURN_SPEC_CONST_HANDLER>, op1
= {constant = 4122498152, var = 4122498152, num = 4122498152, hash =
140737315886184, opline_num = 4122498152, jmp_addr = 0x7ffff5b85468,
zv = 0x7ffff5b85468, literal = 0x7ffff5b85468,
    ptr = 0x7ffff5b85468}, op2 = {constant = 0, var = 0, num = 0,
hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0,
ptr = 0x0}, result = {constant = 0, var = 0, num = 0, hash = 0,
opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0, ptr = 0x0},
  extended_value = 0, lineno = 0, opcode = 62 '>', op1_type = 1
'\001', op2_type = 0 '\000', result_type = 0 '\000'}
```

```
struct _zend_op {
        opcode_handler_t handler;
        znode_op op1;
        znode_op op2;
        znode_op result;
        ulong extended_value;
        uint lineno;
        zend_uchar opcode;
        zend_uchar op1_type;
        zend_uchar op2_type;
        zend_uchar result_type;
};
```

Loaded libraries

```
handler = 0x7ffff4a09280
```

```
(gdb) i shared
From                To                  Syms Read   Shared Object Library
0x00007ffff7fd0100  0x00007ffff7ff25e4  Yes (*)     /lib64/ld-linux-x86-64.so.2
0x00007ffff7fa0720  0x00007ffff7faf11c  Yes         /lib/x86_64-linux-gnu/libresolv.so.2
0x00007ffff7e5c3c0  0x00007ffff7f02f28  Yes         /lib/x86_64-linux-gnu/libm.so.6
0x00007ffff7e48220  0x00007ffff7e49179  Yes         /lib/x86_64-linux-gnu/libdl.so.2
0x00007ffff7cbae50  0x00007ffff7de3cee  Yes (*)     /lib/x86_64-linux-gnu/libxml2.so.2
0x00007ffff7ac0630  0x00007ffff7c3508f  Yes         /lib/x86_64-linux-gnu/libc.so.6
0x00007ffff791a920  0x00007ffff79fe897  Yes (*)     /lib/x86_64-linux-gnu/libicuuc.so.66
0x00007ffff7899280  0x00007ffff78a9e0b  Yes (*)     /lib/x86_64-linux-gnu/libz.so.1
0x00007ffff7873200  0x00007ffff7889e02  Yes (*)     /lib/x86_64-linux-gnu/liblzma.so.5
0x00007ffff5db0040  0x00007ffff5db00f9  Yes (*)     /lib/x86_64-linux-gnu/libicudata.so.66
0x00007ffff5d93ae0  0x00007ffff5da34d5  Yes         /lib/x86_64-linux-gnu/libpthread.so.0
0x00007ffff5c491a0  0x00007ffff5d30e32  Yes (*)     /lib/x86_64-linux-gnu/libstdc++.so.6
0x00007ffff5b935e0  0x00007ffff5ba4045  Yes (*)     /lib/x86_64-linux-gnu/libgcc_s.so.1
0x00007ffff7fc2680  0x00007ffff7fc570d  Yes         vld.so
0x00007ffff4a03370  0x00007ffff4a14c13  Yes (*)     ixed.5.4.lin
```

Debugger: JMP Handler

```
(gdb) b *0x7ffff4a09280
Breakpoint 2 at 0x7ffff4a09280
(gdb) c
Continuing.

Breakpoint 2, 0x00007ffff4a09280 in ?? () from
/usr/local/lib/php/extensions/no-debug-non-zts-20100525/ixed.5.4.lin
```

Enter the SG JMP Handler

```
(gdb) disas $rip,$rip+128
Dump of assembler code from 0x7ffff4a09280 to 0x7ffff4a09300:
=> 0x00007ffff4a09280: push   rbp
   0x00007ffff4a09281: movabs rsi,0xaaaaaaaaaaaaaaab
   0x00007ffff4a0928b: push   rbx
   0x00007ffff4a0928c: sub    rsp,0x8
   0x00007ffff4a09290: mov    rdx,QWORD PTR [rip+0x210ff9]
   0x00007ffff4a09297: mov    rbx,QWORD PTR [rdi]
   0x00007ffff4a0929a: mov    rax,QWORD PTR [rdi+0x28]
   0x00007ffff4a0929e: movsxd rdx,DWORD PTR [rdx]
   0x00007ffff4a092a1: mov    rbp,QWORD PTR [rbx+0x8]
   0x00007ffff4a092a5: mov    rcx,rbp
   0x00007ffff4a092a8: mov    rdx,QWORD PTR [rax+rdx*8+0xd0]
   0x00007ffff4a092b0: mov    rax,QWORD PTR [rax+0x40]
   0x00007ffff4a092b4: sub    rcx,rax
   0x00007ffff4a092b7: mov    rdx,QWORD PTR [rdx]
   0x00007ffff4a092ba: sar    rcx,0x4
   0x00007ffff4a092be: imul   rcx,rsi
   0x00007ffff4a092c2: shl    rcx,0x4
   0x00007ffff4a092c6: mov    ecx,DWORD PTR [rcx+rdx*1]
   0x00007ffff4a092c9: lea    rcx,[rcx+rcx*2]
   0x00007ffff4a092cd: shl    rcx,0x4
   0x00007ffff4a092d1: lea    rcx,[rax+rcx*1]
   0x00007ffff4a092d5: mov    QWORD PTR [rbx+0x8],rcx
   0x00007ffff4a092d9: mov    rcx,rbx
   0x00007ffff4a092dc: sub    rcx,rax
   0x00007ffff4a092df: mov    rax,rcx
   0x00007ffff4a092e2: sar    rax,0x4
   0x00007ffff4a092e6: imul   rax,rsi
   0x00007ffff4a092ea: shl    rax,0x4
   0x00007ffff4a092ee: call   QWORD PTR [rdx+rax*1+0x8]
   0x00007ffff4a092f2: mov    QWORD PTR [rbx+0x8],rbp
   0x00007ffff4a092f6: add    rsp,0x8
   0x00007ffff4a092fa: pop    rbx
   0x00007ffff4a092fb: pop    rbp
   0x00007ffff4a092fc: ret
```

```
0x00007ffff4a092ee: call   QWORD PTR [rdx+rax*1+0x8]
```

```
(gdb) si
ZEND_JMP_SPEC_HANDLER (execute_data=0x7ffff5b4c9e0) at php-
src/Zend/zend_vm_execute.h:430
430 {
```

Jmp_addr changed!

Before entering the SG Handler

```
{handler = 0x7ffff4a09280, op1 = {constant = 4122471032, var = 4122471032, num = 4122471032,
hash = 140737315859064, opline_num = 4122471032, jmp_addr = 0x7ffff5b7ea78, zv =
0x7ffff5b7ea78, literal = 0x7ffff5b7ea78, ptr = 0x7ffff5b7ea78}, op2 = {constant = 0,

    var = 0, num = 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0, ptr =
0x0}, result = {constant = 0, var = 0, num = 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv =
0x0, literal = 0x0, ptr = 0x0}, extended_value = 0, lineno = 0, opcode = 42 '*',

  op1_type = 0 '\000', op2_type = 0 '\000', result_type = 0 '\000'}
```

Before the Zend JMP handler is called

```
{handler = 0x7ffff4a09280, op1 = {constant = 4122479576, var = 4122479576, num = 4122479576,
hash = 140737315867608, opline_num = 4122479576, jmp_addr = 0x7ffff5b80bd8, zv =
0x7ffff5b80bd8, literal = 0x7ffff5b80bd8, ptr = 0x7ffff5b80bd8}, op2 = {constant = 0,

    var = 0, num = 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv = 0x0, literal = 0x0, ptr =
0x0}, result = {constant = 0, var = 0, num = 0, hash = 0, opline_num = 0, jmp_addr = 0x0, zv =
0x0, literal = 0x0, ptr = 0x0}, extended_value = 0, lineno = 0, opcode = 42 '*',

  op1_type = 0 '\000', op2_type = 0 '\000', result_type = 0 '\000'}
```

SourceGuardian JMP Handler

handle_jmp(* zend_execute_data)

1. Current operation (opline) is referenced inside zend_execute_data.

```
struct _zend_execute_data {
        struct _zend_op *opline;
```

2. Operands for current operation are deobfuscated.

```
struct _zend_op {
        znode_op op1;           // fixed!
```

3. Default implementation of the Zend VM's JMP handler is then called with valid operands.

```
ZEND_SPEC_JMP_HANDLER(* zend_execute_data)
```

4. Operands of current operation are restored to their obfuscated values.

```
struct _zend_op {
        znode_op op1; // obfuscated again
```

PHP - SourceGuardian

Solution: Fix each zend_op

fix_jmp(* zend_execute_data)

1. Current operation (opline) is referenced inside zend_execute_data.

```
struct _zend_execute_data {
        struct _zend_op *opline;
```

2. Operands for current operation are deobfuscated.

```
struct _zend_op {
        znode_op op1;                    // fixed!
```

3. Zend VM's JMP handler is called with valid operands

```
ZEND_SPEC_JMP_HANDLER(* zend_execute_data)
```

+ Zend_op's handler address is changed from SG address to ZEND_SPEC_JMP_HANDLER

```
opline->handler = ZEND_SPEC_JMP_HANDLER
```

5. Operands of current operation are restored to their obfuscated values

```
struct _zend_op {
        znode_op op1; // obfuscated again
```

Solution: Fix The Op Array

Before

Op array is dumped, showing obfuscated values.

When an SG handler runs for a zend_op, operands are deobfuscated, zend handler executes, and operands are obfuscated again.

After

Using modified logic from SG handlers… the zend_op's are allowed to deobfuscate as intended. The op handler is set to the zend handler.

"Fixed" zend_op's are dumped with corrected operands.

Zend_ops are executed with corrected operands by the Zend handler.

5 SourceGuardian Opcode Handlers

```
switch (execute_data->op_array->opcodes[i].opcode)
{
        // 42
        case ZEND_JMP:
        // 100
        case ZEND_GOTO:
                fix_jmp(execute_data, sg_offset);
                break;
        // 46
        case ZEND_JMPZ_EX:
        // 47
        case ZEND_JMPNZ_EX:
        // 152
        case ZEND_JMP_SET:
        // 158
        case ZEND_JMP_SET_VAR:
                fix_jmpnz_ex(execute_data, sg_offset)
                break;
        // 45
        case ZEND_JMPZNZ:
                fix_jmpznz(execute_data, sg_offset);
                break;
        // 68
        case ZEND_NEW:
        // 78
        case ZEND_FE_FETCH:
        // 77
        case ZEND_FE_RESET:
                fix_new(execute_data, sg_offset);
                break;
        // 107
        case ZEND_CATCH:
                fix_catch(execute_data, sg_offset);
                break;
        default:
                break;
}
```

- 42 JMP
- 100 GOTO

- 46 JMPZ_EX
- 47 JMPNZ_EX
- 152 JMP_SET
- 158 JMP_SET_VAR

- 45 JMPZNZ

- 68 NEW
- 78 FE_FETCH
- 77 FE_RESET

- 107 CATCH

Classes and functions

```
class ClassOne
{
    function func_one()
    {
        echo "one";
    }


function notused_one()
    {
        return 1;
    }
}
```

```
class ClassTwo
{
    function func_two()
    {
        echo "two";
    }


function notused_two()
    {
        return 2;
    }
}
```

```
$a = rand(1, 2);


if ($a == 1)
{
    $b = new ClassOne();
    $b->func_one();
}


else
{
    $b = new ClassTwo();
    $b->func_two();
}
```

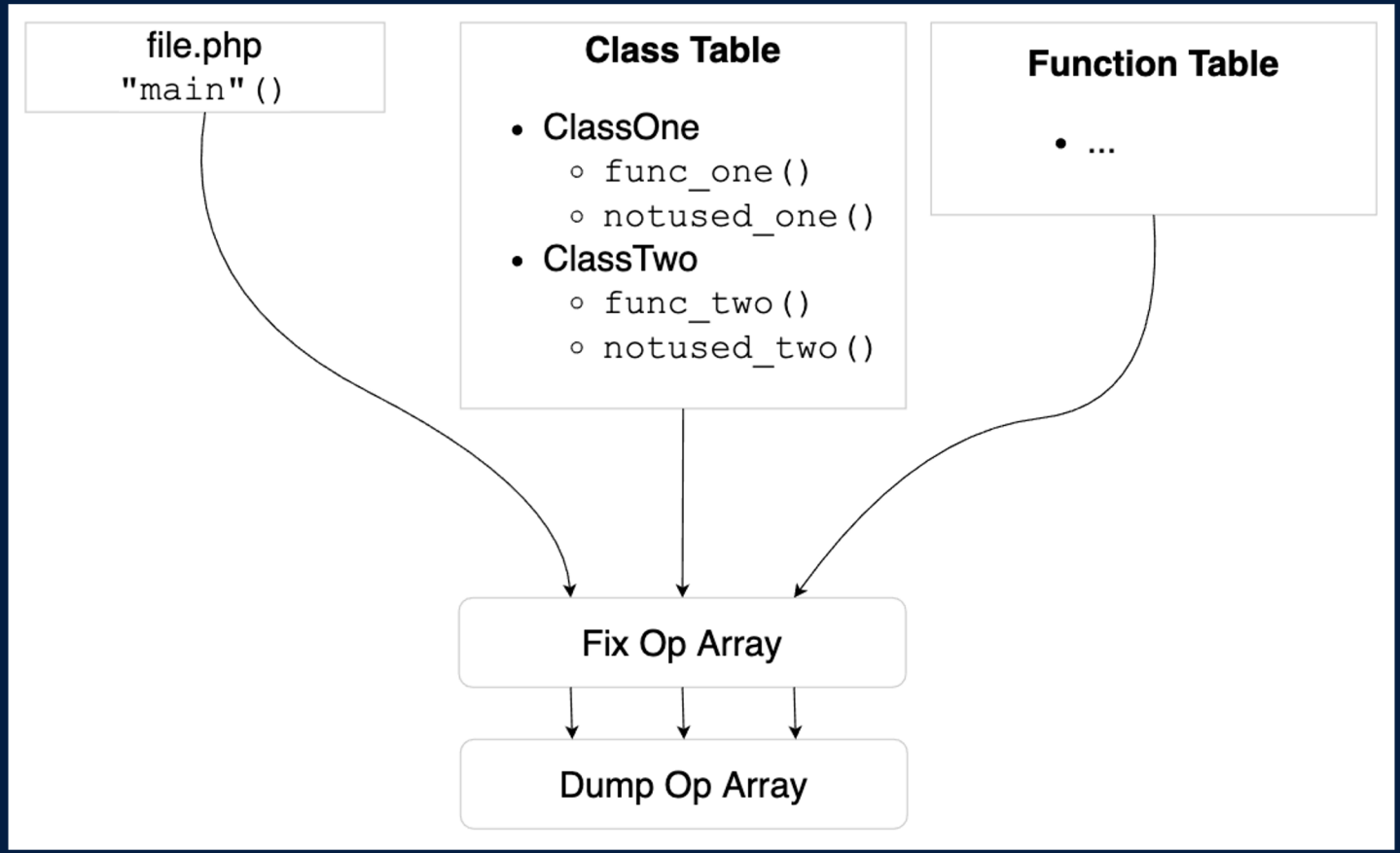
Classes and functions: “main”()

```
execute()
filename:       class.php
function name:  (null)
number of ops:  24
compiled vars:  !0 = $a, !1 = $b
line     #* E I O op                           fetch          ext  return  operands
-------------------------------------------------------------------------------
   0     0*       JMP                                                    OP1[ ->19 ]
         1*       RETURN                                                 OP1[  IS_CONST (114731043) 1 ]
         2*       NOP
         3*       NOP
         4*       SEND_VAL                                               OP1[  IS_CONST (114731025) 1 ]
         5*       SEND_VAL                                               OP1[  IS_CONST (114731027) 2 ]
         6*       DO_FCALL                                 2     RES[  IS_VAR $2 ]     OP1[  IS_CONST (114731028) 'rand' ]
         7*       ASSIGN                                         RES[  ]         OP1[  IS_CV !0 ] OP2[ ,  IS_VAR $2 ]
         8*       IS_EQUAL                                       RES[  IS_TMP_VAR ~4 ]       OP1[  IS_CV !0 ] OP2[ ,  IS_CONST (114731029) 1 ]
         9*       JMPZNZ                                  17             OP1[  IS_TMP_VAR ~4 ] OP2[ , ->0 ]
        10*       FETCH_CLASS                              4     RES[ :5 ]       OP2[  IS_CONST (114731030) 'ClassOne' ]
        11*       NEW                                            RES[  ]         OP1[ :5 ]
        12*       DO_FCALL_BY_NAME                         0     RES[  ]         OP1[  ]
        13*       ASSIGN                                         RES[  ]         OP1[  IS_CV !1 ] OP2[ ,  IS_VAR $6 ]
        14*       INIT_METHOD_CALL                               RES[  ]         OP1[  IS_CV !1 ] OP2[ ,  IS_CONST (114731034) 'func_one' ]
        15*       DO_FCALL_BY_NAME                         0     RES[  ]         OP1[  ]
        16*       JMP                                                    OP1[ ->23 ]
        17*       FETCH_CLASS                              4     RES[ :11 ]      OP2[  IS_CONST (114731037) 'ClassTwo' ]
        18*       NEW                                            RES[  ]         OP1[ :11 ]
        19*       DO_FCALL_BY_NAME                         0     RES[  ]         OP1[  ]
        20*       ASSIGN                                         RES[  ]         OP1[  IS_CV !1 ] OP2[ ,  IS_VAR $12 ]
        21*       INIT_METHOD_CALL                               RES[  ]         OP1[  IS_CV !1 ] OP2[ ,  IS_CONST (114731040) 'func_two' ]
        22*       DO_FCALL_BY_NAME                         0     RES[  ]         OP1[  ]
        23*       JMP                                                    OP1[ ->23 ]
```

Classes and functions: func_one()

```
execute()
filename:       class.php
function name:  func_one
number of ops:  3
compiled vars:  none
line     #* E I O op                           fetch          ext  return  operands
---------------------------------------------------------------------------------
         0*       JMP                                                    OP1[ ->2 ]
         1*       ECHO                                                   OP1[  IS_CONST (114730970) 'one' ]
         2*       RETURN                                                 OP1[  IS_CONST (114730971) null ]

one
```

Dump it All!


file.php
"main"()
Class Table
ClassOne
func_one()
notused_one()
ClassTwo
func_two()
notused_two()
Function Table
...
Fix Op Array
Dump Op Array

Dump it All - ClassOne

```
Class ClassOne:
Function func_one:
filename:       class_one.php
function name:  func_one
number of ops:  3
compiled vars:  none
line     #* E I O op                           fetch          ext  return  operands
---------------------------------------------------------------------------------
         0*       JMP                                                    OP1[ ->1 ]
         1*       ECHO                                                   OP1[  IS_CONST (50714171) 'one' ]
         2*       RETURN                                                 OP1[  IS_CONST (50714172) null ]

End of function func_one

Function notused_one:
filename:       class_one.php
function name:  notused_one
number of ops:  2
compiled vars:  none
line     #* E I O op                           fetch          ext  return  operands
---------------------------------------------------------------------------------
         0*       JMP                                                    OP1[ ->1 ]
         1*       RETURN                                                 OP1[  IS_CONST (50714191) 1 ]

End of function notused_one

End of class ClassOne.
```

Dump it All - ClassTwo

```
Class ClassTwo:
Function func_two:
filename:       class_one.php
function name:  func_two
number of ops:  3
compiled vars:  none
line     #* E I O op                                fetch          ext  return  operands
-------------------------------------------------------------------------------------
         0*       JMP                                                      OP1[ ->1 ]
         1*       ECHO                                                     OP1[  IS_CONST (50714234) 'two' ]
         2*       RETURN                                                   OP1[  IS_CONST (50714235) null ]

End of function func_two

Function notused_two:
filename:       class_one.php
function name:  notused_two
number of ops:  2
compiled vars:  none
line     #* E I O op                                fetch          ext  return  operands
-------------------------------------------------------------------------------------
         0*       JMP                                                      OP1[ ->1 ]
         1*       RETURN                                                   OP1[  IS_CONST (50714320) 2 ]

End of function notused_two

End of class ClassTwo.

Two
```

Thanks for joining!

Blogs on Tenable TechBlog:

- Remapping Python Opcodes
- Dumping PHP Opcodes Protected by SourceGuardian

Code:

- https://github.com/tenable/poc/blob/master/druva/inSync/convert_pyc_opcodes.py
- https://github.com/tenable/vld-sourceguardian