Hacking Python 🐍 Apps

Suraj (s0cket7)

About s0cket7

DREAM11

Security Researcher

About s0cket7

Active CTF-er

About s0cket7

hackerone

Bug Bounty Hunter

Agenda

- Introduction
- 🐍 Security
- Exploiting 🐍 Apps
- Demos 😁
- Mitigations 🔨

Introduction to 🐍

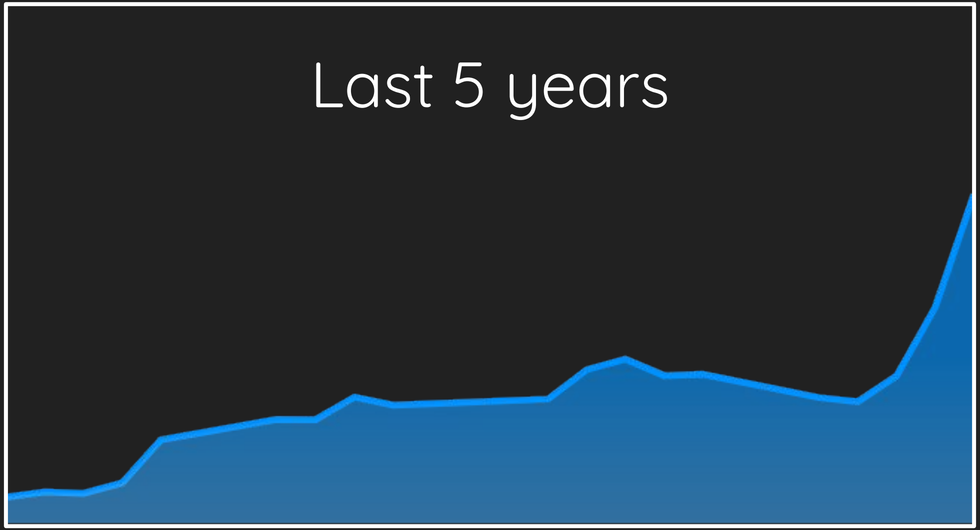
Interpreted, Simple and Elegant

```
/* Java */
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}


# Python
print("Hello, World!")
```

Rapid Development 💨

```
# Convert “Hello, World!” to it’s hexadecimal equivalent
''.join([hex(ord(i)).replace('0x','') for i in "Hello, World!"])
# '48656c6c6f2c20576f726c6421'


# Even Better
"Hello, World!".encode('hex')
# '48656c6c6f2c20576f726c6421'
```

🐒 Evolution 🤵



Community 😃

Modules for everything and PyCon

Used everywhere ⚡

🐍 Security

Memory Leaks 📜

Format String Bugs and many others

Arbitrary Memory Write ✍️

Buffer Overflows on Stack & Heap

Web 🕸️

Frameworks like Django Protects against XSS, CSRF, SQLi and other issues.

But Python has it’s quirks…

Let’s setup the environment

Let’s Hack 🐍 Applications

Code Execution 😱

```
os.system(user_input)
```

or

```
eval(user_input)
```

Classic case

```
ip_address = input("Enter IP: ")
os.system("ping " + ip_address)

# ping google.com

# ping google.com;my_command
```

🧑‍💻 Demo?

What’s wrong ? 🤔

```python
#!/usr/bin/python
number = input("What is 1*1337?")
print("your answer is", number)
```

What’s wrong ? 🤔

```
#!/usr/bin/python3
number = input("What is 1*1337?")
print("your answer is", number)
```

Python3 🐍🐍🐍

```
input("What is 1*1337?")
```

Python2 🐍🐍

```
raw_input("What is 1*1337?")
```

What is `input()` in Python2?

```
eval(raw_input("What is 1*1337?"))
```

🧑‍💻 Demo?

Mitigations 🔨

1 . Don’t execute shell commands

2 . Don’t pass the user input

3 . Proper sanitization

Optimization Bugs 🔧

```
assert 1==1

assert 1==2
# Assertion Error

assert security_level==1
# Never do this, why?
```

Optimization Bugs 🔧

```
$ python -c "assert 1==2"
Traceback (most recent call last):
  File "<string>", line 1, in <module>
AssertionError


$ python -O -c "assert 1==2"
  ‘-O’ is the optimization flag
```

🧑‍💻 Demo?

Mitigations 🔨

1. Don't use `assert`

2. I can't say "Don't optimize" 😅

Bad Import 🎃

How modules are imported in python?

```
import module
```

Bad Import 🎃

1 . Builtins

2 . Cache/Imported

3 . Current directory

4 . PYTHONPATH

5 . Default installation

Bad Import 🎃

```
import some_module

some_module.some_function("some data")

some_module => ./some_module.py
```

🧑‍💻 Demo?

Mitigations 🔨

1 . Do not provide extra privileges

2 . If privileges are necessary, import Modules via `imp` or `importlib` using the full path.

Types Mismatch

```
import json

data = '{"count": 10}'
json_data = json.loads(data)

if (json_data['count'] > 12):
    print('Got em!')
else:
    print('Nope!')
```

Types Mismatch

```
import json

data = '{"count": 10}'
json_data = json.loads(data)

if (json_data['count'] > 12):
    print('Got em!')
else:
    print('Nope!')
```

Types in Python2

```
>>> 1 > 10
False
```

```
>>> "1" > 10
True
```

Why this behaviour?

“Objects of different types except numbers are ordered by their type names; objects of the same types that don’t support proper comparison are ordered by their address.”

```
numbers -> dictionary -> list -> str -> tuple
```

Types Mismatch

```
import json

data = '{"count": 10}'
json_data = json.loads(data)

if (json_data['count'] > 12):
   print('Got em!')
else:
   print('Nope!')
```

Types Mismatch

```
import json

data = '{"count": "10"}'
json_data = json.loads(data)

if (json_data['count'] > 12):
    print('Got em!')
else:
    print('Nope!')
```

👨‍💻 Demo?

Mitigations 🔨

1 . Use Python3 instead.

2 . Check the type or convert to a specific type you are expecting and if there’s an exception, raise it.

Deserialization ❤️

What is serialization?

Some data $\longrightarrow$ Stream of bytes

Serialization Formats

1 . Native

2 . Generic

3 . Special

Builtin modules 🗜️

1 . json

2 . marshal

3 . pickle

JSON

```json
{
    "name": "JaSON",
    "age": 22,
    "hobbies": [
        "Basketball",
        "Painting"
    ],
    "address": {
        "Street": "Wut St",
        "House No": 1337
    },
    "isStudent": true,
    "badHabbits": null
}
```

JSON

Use case?

API

MARSHAL

Internal Object Serialization

compiled code $\longrightarrow$ bytes

MARSHAL

Use case?

Internal use only

PICKLE

Serialize objects $\longrightarrow$ bytes

PICKLE

```
# serialization
some_object = SomeClass()
serialized = pickle.dumps(some_object)


# deserialization
new_some_object = pickle.loads(serialized)
```

PICKLE

Use case?

Machine Learning, Cookies …

PICKLE

What can go wrong?

`__reduce__()`

`__reduce__()`

Automatically invoked for hints &
Returns str/tuple

Recap

1. Serialize
2. `__reduce__`
3. Auto invoke

So.....

If I have an object with `__reduce__` method, when unserialized, we get code execution?

👨‍💻 Demo?

Mitigations 🔨

1. Don’t use pickle

2. Don’t deserialize user input

3. If you want to have serialization for some reason, sign it.

Conclusion 🔨

1 . Don’t reinvent security

2 . Use tested packages

3 . Read Documentation

4 . Update yourself with security

🙅‍♂️Don’t Ignore Security

@s0cket7

s0cket7.com

ありがとう

Thank you