E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Título del Trabajo Fin de Grado

Grado en Ingeniería Informática

Trabajo Fin de Grado

Iban Ruiz de Galarreta Cadenas

Santiago García Jiménez

Pamplona, 05/09/2024

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Contenido

1. RESUMEN

El proyecto desarrollado se centra en una aplicación que crea máquinas virtuales web con Docker para la práctica de hacking ético y pruebas de penetración, con énfasis en la educación y entrenamiento en ciberseguridad.

El objetivo es crear máquinas web con vulnerabilidades, proporcionando un entorno de aprendizaje realista y desafiante. Estas máquinas generan vulnerabilidades de manera aleatoria, imitando la naturaleza cambiante de las amenazas cibernéticas. Docker se utiliza por su capacidad de virtualización y seguridad para crear entornos controlados.

El proyecto se destaca por su flexibilidad y capacidad de integración de nuevas características, permitiendo la expansión continua con nuevas vulnerabilidades y configuraciones. Además, se subraya la importancia de documentar experiencias a través de write-ups personalizados, lo que enriquece el aprendizaje y la comprensión de cómo abordar y mitigar vulnerabilidades.

El funcionamiento de la herramienta se adapta a dos modalidades: una para usuarios que practican directamente y otra para la creación y gestión de múltiples máquinas, ideal para plataformas de CTF como HackTheBox. En resumen, este proyecto busca fortalecer las habilidades de ciberseguridad de los participantes y contribuir a la defensa contra amenazas cibernéticas en constante evolución.

2. INRODUCCIÓN

En la era actual, marcada por la interconexión global y la creciente digitalización, la ciberseguridad se ha convertido en un componente esencial para garantizar la estabilidad y confiabilidad de sistemas informáticos. A medida que la sociedad depende cada vez más de la tecnología, la amenaza de ciberataques se vuelve más prominente, lo que destaca la importancia de contar con profesionales capacitados en seguridad informática.

Noticias recientes han subrayado la urgencia de abordar las vulnerabilidades cibernéticas. Incidentes de alto perfil, como brechas de datos en grandes corporaciones, ataques ransomware a infraestructuras críticas y actividades cibernéticas maliciosas respaldadas por estados, han puesto de manifiesto la necesidad crítica de fortalecer las defensas cibernéticas. Como ejemplo de estas noticias tenemos el reciente ransomware (Lockbit 3.0) que ha afectado a la guardia civil y a las fuerzas armadas comprometiendo millones de datos personales y médicos.

La llegada de las IAs ha aumentado la productividad de los programadores, facilitando el desarrollo de las aplicaciones. Sin embargo, esto también ayuda a los ciberdelincuentes, dotándolos de mayores capacidades para crear código malicioso con menores conocimientos, lo que ha aumentado notablemente los ataques de ransomware. Otra noticia que demuestra el crecimiento del cibercrimen es el informe de la agencia de Ciberseguretat de Catalunya gestionó más de 5.000 millones de ciberataques en 2023. Estas noticias se pueden encontrar en el apartado de referencias.

En este contexto, los hackers éticos desempeñan un papel fundamental. A diferencia de los hackers maliciosos, los hackers éticos emplean sus habilidades para identificar y corregir vulnerabilidades, contribuyendo así a fortalecer la seguridad digital. Plataformas de formación y práctica, como HackTheBox, TryHackMe y similares, ofrecen entornos seguros para que los profesionales y entusiastas de la ciberseguridad perfeccionen sus habilidades a través de desafíos realistas y escenarios simulados.

El proyecto que estamos desarrollando, centrado en la creación de máquinas web con Docker que simulan entornos de servidores vulnerables, se alinea con esta necesidad creciente de entrenamiento práctico en ciberseguridad. Al proporcionar un entorno controlado para simular situaciones de amenaza, nuestro proyecto no solo sirve como una herramienta educativa valiosa, sino que también aborda directamente la necesidad de adoptar un enfoque proactivo hacia la seguridad informática, permitiendo a los

participantes comprender distintos tipos de vulnerabilidades, para tener una mejor visión de como asegurar sistemas antes de enfrentarse a situaciones del mundo real.

Comparativamente, plataformas como HackTheBox y TryHackMe comparten el objetivo común de ofrecer desafíos y entornos de práctica para mejorar las habilidades en ciberseguridad. Cada una tiene su enfoque único y su comunidad activa, y nuestro proyecto busca complementar estas iniciativas existentes al proporcionar una alternativa centrada en máquinas web con Docker, ampliando así las opciones de entrenamiento disponible para la comunidad de ciberseguridad. Esta aplicación desempeña un papel crucial en la formación de profesionales altamente competentes que desempeñarán un papel vital en la defensa contra las amenazas cibernéticas en evolución constante.

1.1. Un Terreno de Juego Educativo: Hacking Ético y Pruebas de Penetración

La ciberseguridad no solo se trata de defenderse contra las amenazas, sino también de comprender cómo funcionan y, en consecuencia, fortalecer las defensas. En este contexto, la creación de máquinas web con vulnerabilidades variadas proporciona un terreno de juego educativo para el hacking ético y las pruebas de penetración. Al simular escenarios del mundo real con amenazas específicas, los desarrolladores y profesionales de la seguridad pueden perfeccionar sus habilidades en la detección y mitigación de vulnerabilidades. Este proyecto crea un entorno en el que se puede tanto diseñar como explotar distintos tipos de vulnerabilidades web lo que crea un entorno educativo y entretenido para la educación sobre ciberseguridad.

1.2. La Dinámica de Vulnerabilidades Cambiantes: Adaptación Continua

Una de las características distintivas de este proyecto radica en la generación aleatoria de vulnerabilidades en cada instancia creada. Este enfoque imita la realidad en la que las amenazas informáticas evolucionan constantemente. Al enfrentarse a diferentes desafíos en cada despliegue, los practicantes adquieren nuevos conocimientos y no se centran en automatizar procesos. Aprendiendo a adaptarse a los distintos entornos.

1.3. Docker: Virtualización y Aislamiento para una Mayor Seguridad

La elección estratégica de Docker como plataforma para la creación de máquinas virtuales no es casual. Docker, al centrarse en la virtualización de contenedores, simplifica la generación eficiente de entornos reproducibles. Sin embargo, es crucial señalar que el propósito de este proyecto no radica en atacar infraestructuras generadas en Docker, sino en la utilización de Docker como una herramienta eficaz para la creación de entornos virtualizados.

1.3. La Importancia Crucial de la Seguridad Web

La seguridad web se ha convertido en un aspecto vital en un mundo interconectado. Siendo la web uno de los principales servicios utilizados por las empresas y normalmente expuesto al exterior, la protección de las aplicaciones web se vuelve esencial. Este proyecto no solo busca proporcionar un entorno educativo para comprender y explotar las vulnerabilidades en servidores web, sino que también subraya la importancia de implementar prácticas robustas de seguridad web desde la fase inicial del desarrollo. La seguridad web no solo protege los datos y servicios en línea, sino que también fortalece la confianza de los usuarios y la integridad de las organizaciones que operan en el vasto ecosistema digital.

1.4. Elección del Lenguaje C: Optimización y Control en la Orquestación con Docker

La elección de utilizar el lenguaje C en el desarrollo de este proyecto se fundamenta en el hecho de que es el lenguaje en el que se posee mayor dominio y experiencia. Esta elección permite aprovechar al máximo las capacidades del lenguaje para implementar el programa de manera eficiente y con un alto grado de control sobre los recursos del sistema. Además, al ser un lenguaje ampliamente utilizado en la carrera y en el campo de la programación en general, facilita la comprensión y el manejo de los aspectos críticos relacionados con la seguridad informática y la gestión de vulnerabilidades en el entorno de máquinas web.

3. OBJETIVOS

El proyecto tiene como objetivo principal crear un entorno de aprendizaje dinámico y desafiante para la ciberseguridad, centrado en el desarrollo de máquinas web simuladas con vulnerabilidades. Este entorno debe ser ampliable, de tal forma que se pueda ampliar fácilmente con cualquier otra vulnerabilidad, base de datos, sistema… Estos objetivos se desglosan en los siguientes aspectos:

1.2. Aleatoriedad Integral: Simulando Escenarios del Mundo Real

Este proyecto genera las máquinas web de forma aleatoria, no solo las vulnerabilidades, sino también la apariencia de la web, las bases de datos y los datos almacenados en ellas. Esta variabilidad refleja la diversidad de las máquinas que puedes encontrar en entornos del mundo real, proporcionando a los participantes una experiencia de aprendizaje práctica y real.

1.3. Vulnerabilidades Web: Ejecución Remota y Elevación de Privilegios

El componente central del proyecto se enfoca en la creación de máquinas web simuladas con ciertas vulnerabilidades. Estas vulnerabilidades se dividen en dos categorías:

- **Ejecución Remota de la Máquina:** A través de distintos fallos web, los participantes deberán identificar y explotar vulnerabilidades que les permitan lograr la ejecución remota de la máquina. Esta tarea implica la capacidad de manipular la aplicación web para obtener acceso y control sobre el sistema.
- **Elevación de Privilegios para Administrador:** La segunda categoría de vulnerabilidades implica el desafío de elevar los privilegios para obtener control total sobre la máquina, adquiriendo privilegios de administrador. Aunque el proyecto implementa esta faceta, se enfocará principalmente en la primera parte, proporcionando un énfasis especial en las técnicas de ejecución remota.

Estos objetivos buscan no solo fortalecer las habilidades técnicas en la identificación y explotación de vulnerabilidades, sino también fomentar una comprensión profunda de la importancia de implementar prácticas seguras en el desarrollo de aplicaciones web.

1.4. Importancia de Write-ups Personalizados: Documentando Experiencias y Mejorando Aprendizaje

Adicionalmente, el proyecto tiene como objetivo la creación de write-ups personalizados para cada máquina simulada. Estos documentos detallarán una forma de superar las vulnerabilidades específicas presentes en cada escenario. La elaboración de write-ups no solo servirá como una valiosa fuente de documentación, sino que también contribuirá al proceso educativo al proporcionar un recurso de aprendizaje adicional.

Los write-ups ofrecerán una visión detallada de los métodos utilizados para identificar, explotar y mitigar las vulnerabilidades, brindando a los participantes una guía práctica y contextualizada. Al fomentar la documentación detallada de cada experiencia, el proyecto busca promover un enfoque reflexivo y analítico en la resolución de problemas de ciberseguridad, permitiendo a los participantes interiorizar y aplicar los conocimientos adquiridos.

Se espera que la inclusión de esta práctica contribuya significativamente a la mejora continua de las habilidades de ciberseguridad de los participantes y fortalezca la conciencia sobre las mejores prácticas en el ámbito de la seguridad informática.

1.5. Facilidad de Integración de Nuevas Características en el Generador de CTFs

Una de sus fortalezas radica en la facilidad para integrar nuevas características, tanto estéticas como funcionales, así como en la incorporación de nuevas vulnerabilidades y configuraciones de máquina. Esta flexibilidad es crucial para mantener la relevancia del proyecto en un entorno de ciberseguridad en constante cambio, donde emergen nuevas amenazas y se requieren herramientas actualizadas para formación y evaluación.

Facilidad de Integración

El generador de CTFs ha sido diseñado con una estructura que permite la incorporación de nuevas funcionalidades sin necesidad de realizar modificaciones significativas en el código base. Esto ha sido posible gracias a una documentación detallada que guía el proceso de expansión del sistema. Tanto las características estéticas como las configuraciones de máquina y las vulnerabilidades pueden añadirse de manera ágil, manteniendo la coherencia y estabilidad del proyecto.

Incorporación de Nuevas Vulnerabilidades

El sistema facilita la adición de nuevas vulnerabilidades, permitiendo a los desarrolladores incluir fallos de seguridad recientes, como inyecciones SQL, fallos de configuración o escalada de privilegios, sin alterar el núcleo del proyecto. Esta capacidad de expansión asegura que el generador de CTFs se mantenga actualizado frente a las últimas amenazas en ciberseguridad, ofreciendo escenarios relevantes y actuales para los usuarios.

Configuración de la Máquina y Personalización Estética

Además de las vulnerabilidades, es posible modificar o añadir nuevas configuraciones de máquina y características estéticas de manera sencilla. La documentación proporciona instrucciones claras para personalizar estos aspectos, permitiendo adaptar el entorno del CTF a necesidades específicas o preferencias del usuario, lo que mejora la experiencia y relevancia de los desafíos generados.

Ventajas de la Documentación Detallada

1. **Facilidad de Mantenimiento:** La existencia de una documentación detallada facilita el mantenimiento y actualización del sistema, permitiendo que se realicen mejoras sin comprometer la estabilidad del proyecto.

2. **Escalabilidad:** A medida que surgen nuevas necesidades en el campo de la ciberseguridad, es posible escalar el sistema con nuevas características y vulnerabilidades, sin necesidad de restructuraciones profundas.

3. **Colaboración:** La clara documentación fomenta la colaboración entre diferentes desarrolladores o equipos, permitiendo que cada uno pueda trabajar en la adición de nuevas funcionalidades de forma independiente y eficiente.

4. **Personalización:** Los usuarios pueden adaptar el generador de CTFs a sus necesidades específicas gracias a la facilidad de integrar nuevas características y configuraciones, permitiendo un enfoque más dirigido en la formación o evaluación de competencias.

3. FUNCIONAMIENTO

Esta herramienta ha sido diseñada para ofrecer flexibilidad dependiendo de las necesidades, puede ser utilizada de dos formas distintas: una orientada a la práctica directa del alumno y otra para la creación y gestión de múltiples máquinas vulnerables. A continuación, se detallan ambas modalidades de uso.

3.1. Uso de la Herramienta en Modo Alumno

La herramienta ofrece una interfaz simple y directa para los alumnos, permitiéndoles generar y trabajar con máquinas vulnerables de manera inmediata. Al ejecutar el programa desde la consola, se crea una máquina virtual que se ejecuta por defecto en el puerto 80 (*Imagen 1*), salvo que se especifique otro puerto mediante los parámetros correspondientes. Esta máquina contiene vulnerabilidades que los usuarios deben explotar para obtener dos flags clave:

1. **Flag de Acceso Remoto**: Se obtiene al lograr acceso remoto a la máquina.
2. **Flag de Acceso Privilegiado**: Se consigue cuando se adquiere control total y privilegios elevados dentro del sistema.

Una vez obtenidas estas flags, los usuarios pueden introducirlas mediante el parámetro -f para completar la tarea asignada y, automáticamente, la máquina será eliminada (*Imagen 2*). Si el usuario no desea continuar, puede optar por cancelar la máquina antes de completar todos los objetivos.

```
❯ ./ctf-generator
Creadndo la nueva maquina...
La maquina esta corriendo en el puerto 80.
```

Imagen 1

```
❯ ./ctf-generator -f 646zjia8ua4td2q12w9z6ykm7z5bjbhv
Enhorabuena!! Has conseguido la flag de User.

Objetivos:
Flag de Usuario Conseguida: Sí
Flag de Root Conseguida: No
```

Imagen 2

Los parámetros disponibles en este modo son los siguientes:

-p o --port PUERTO: Especifica el puerto en el que se ejecutará la máquina. Por defecto, es el puerto 80.

-c o --cancel: Elimina la máquina actual, útil para rendirse antes de completar todas las flags, ya que la máquina se elimina automáticamente al completarlas.

-s o --stop: Detiene la máquina sin eliminarla, permitiendo retomarla más adelante.

-r o --run: Reactiva una máquina previamente detenida.

3.2. Uso Avanzado: Creación y Gestión de Múltiples Máquinas

Para usuarios más avanzados, la herramienta ofrece una segunda forma de uso, ideal para la creación y gestión de múltiples máquinas, lo que es especialmente útil para quienes deseen preparar retos en plataformas de CTF como HackTheBox o TryHackMe.

Mediante el parámetro -m, se pueden generar un número ilimitado de máquinas que se almacenan en la carpeta "dockers" dentro del proyecto. Estas máquinas incluyen todos los archivos necesarios para su ejecución y gestión, lo que facilita su transporte y uso en otros entornos.

```
❯ ./ctf-generator -m 3
Maquina 'mvuln1' creada con éxito.
Maquina 'mvuln2' creada con éxito.
Maquina 'mvuln3' creada con éxito.
```

Imagen 3

Cada máquina creada contiene los siguientes archivos:

1. **start.sh**: Un script que inicia la máquina. Si le añadimos un numero como parámetro a este script podremos modificar el puerto donde se creará esta máquina. Por defecto se creará en un puerto que empiece por 8 seguido del numero de la carpeta.
2. **stop.sh**: Un script que detiene la máquina.
3. **write-up.txt**: Un archivo que documenta la resolución de la máquina, proporcionando un recurso invaluable tanto para creadores como para usuarios.

Este modo avanzado permite a los usuarios no solo trabajar con máquinas individuales, sino también crear y distribuir entornos completos para desafíos de ciberseguridad, ofreciendo flexibilidad y control en la preparación de escenarios personalizados.

3.3. Uso común: siguientes pasos

Una vez que la máquina vulnerable ha sido creada y está en funcionamiento, el siguiente paso es conectarse a ella para comenzar el reto. Este proceso es sencillo y se realiza a través del navegador web.

Si la máquina se está ejecutando en el mismo equipo desde el que se va a acceder, simplemente se debe ingresar 127.0.0.1:puerto en la barra de direcciones del navegador, reemplazando "puerto" con el número especificado al crear la máquina (o el puerto 80 si no se cambió el valor por defecto).

En caso de que la máquina esté alojada en un equipo diferente, se deberá ingresar la dirección IP de ese equipo seguida del puerto en cuestión, por ejemplo, 192.168.1.100:puerto.

Al acceder a esta dirección, se abrirá una página web que contiene un formulario de inicio de sesión. Este será el punto de partida del reto, donde el usuario deberá utilizar sus habilidades para explotar las vulnerabilidades de la máquina, superar los desafíos presentados y avanzar hacia la obtención de las flags.

4. IMPLEMENTACIÓN

En este apartado se va a explicar como esta distribuido el proyecto, para poder entender el código y así facilitar la incorporación de nuevas características y vulnerabilidades. Más adelante se explicará como introducirlas, pero este párrafo es necesario para tener una visión más global del proyecto.

En la siguiente Imagen podemos ver los archivos de la carpeta principal, acompañado de una explicación de cada uno (*Imagen 4*).

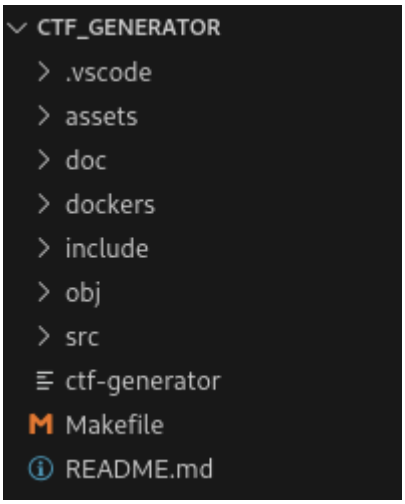


Imagen 4

1. **Assets**: Esta carpeta contiene los archivos necesarios para la creación de las máquinas.
2. **Doc**: Es la documentación necesaria tanto para crear los write-ups como la propia memoria del proyecto.
3. **Dockers**: En esta carpeta se crearán las distintas máquinas, da igual el modo en el que se ejecute.
4. **Include**: Aquí se encuentran los ".h" con las funciones y módulos del proyecto.
5. **Obj**: En esta carpeta se crean los ". o" al compilar el programa con "make".
6. **Src**: Aquí encontramos todo el código de la aplicación, dividida en los distintos módulos que más adelante se explicaran. Su distribución la podemos ver en la *Imagen 5*.
7. **Ctf-generator**: Es el ejecutable de la aplicación
8. **Makefile**: Es un archivo que se encarga de facilitar la compilación

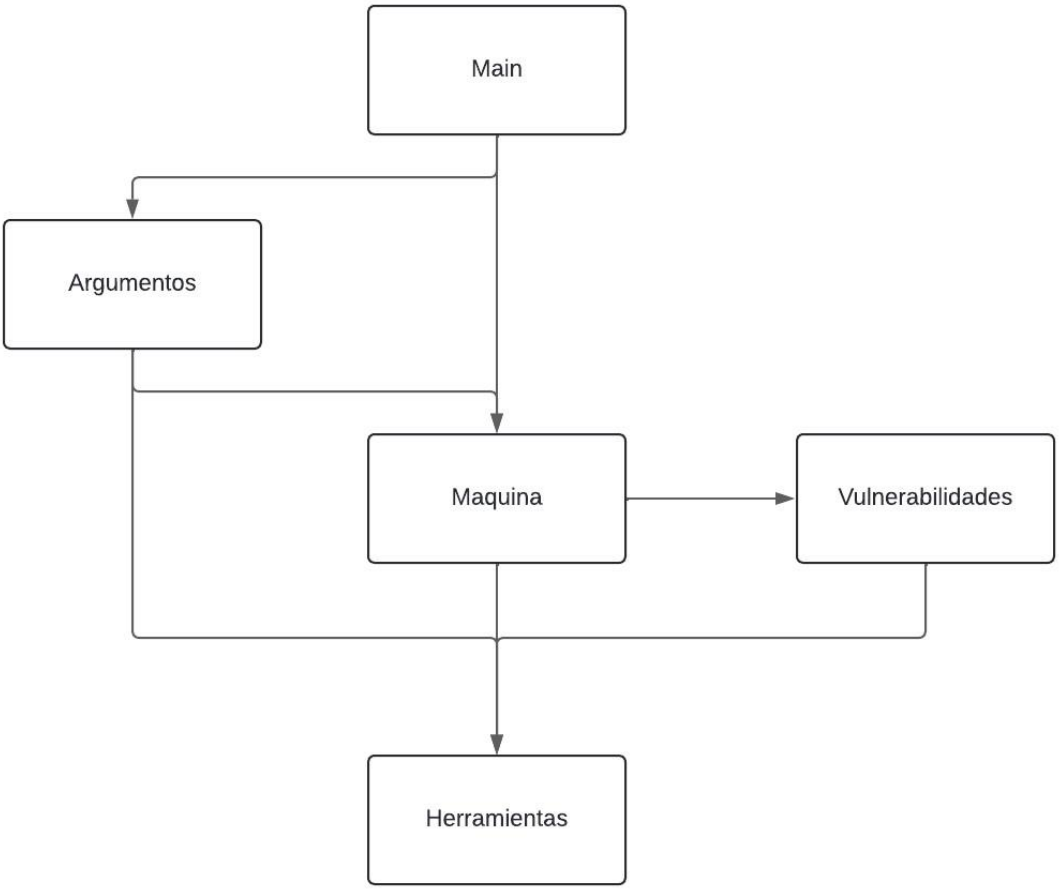
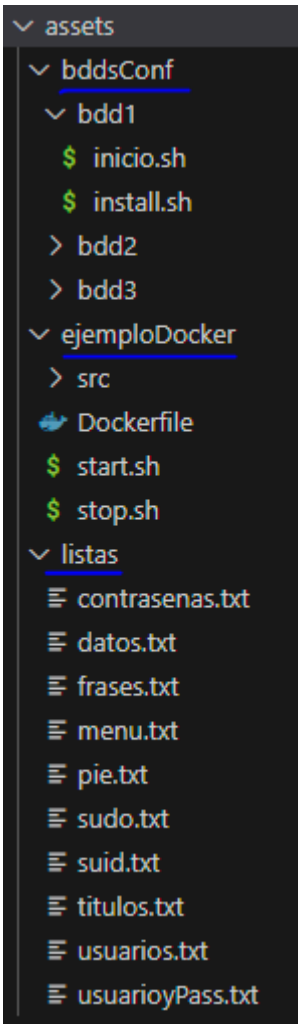


Imagen 5

4.1. Carpeta “assets”

En esta carpeta se encuentran los archivos necesarios para la creación de las maquinas, las carpetas más importantes son “**bddsConf**”, “**listas**”, “**ejemploDocker**” y “**otros**”.

En la primera, ”bddsConf”, hay una carpeta distinta por cada base de datos que haya implementada en el proyecto “bdd1”, “bdd2”, “bdd3” … Dentro de estas habrá, mínimo, dos archivos, uno que inicia los servicios de apache y la base de datos que toque llamado “inicio.sh”. Y otro llamado “install.sh” que contendrá los comandos necesarios que la maquina tiene que ejecutar para instalar esa base de



datos. Este archivo esta sin acabar, ya que el script se encarga de terminar el archivo con datos aleatorios.

En "**listas**" se ubica una serie de listas en la que cada línea tiene un dato, para que el script elija una línea aleatoria y la implemente. Por ejemplo, podemos encontrar las listas "contraseñas.txt" y "usuarios.txt" que contienen distintas contraseñas y distintos usuarios para que sean elegidos aleatoriamente, o "sudo.txt" que contiene la ruta de todos los archivos vulnerables por sudo que contiene la máquina.

"**ejemploDocker**" es un ejemplo del archivo de una máquina, pero sin acabar. Esta carpeta se copiará y pegará en la carpeta "dockers" y a partir de ahí se ira modificando con la configuración. Consta de dos archivos "start.sh" y "stop.sh" que inician y paran la máquina, en estos archivos el script establece un puerto y un nombre distinto para cada máquina. También se puede especificar el puerto pasándoselo a "start.sh" como argumento. A parte estará el archivo "Dockerfile" que contiene la configuración necesaria para crear la maquina y la carpeta "src" con los archivos web.

"**otros**" contiene información necesaria para crear las distintas vulnerabilidades.

Imagen 6

4.2. Modulo "main"

Este es el módulo principal, tiene una serie de Condiciones para leer los distintos argumentos y llamar a las funciones correspondientes. Al principio comprueba que solo se le pasen 2 argumentos como mucho, ya que cada argumento es independiente. Al final comprueba que los requisitos sean correctos (como por ejemplo tener Docker instalado) y que no haya una maquina ya corriendo y en ese caso la crea.

4.3. Modulo "argumentos"

Este módulo recoge las funciones necesarias para el correcto funcionamiento de los argumentos pasados al programa. Se describirá brevemente cada una de estas funciones:

- "**mostrarAyuda**": Saca por pantalla la información sobre los argumentos.
- "**cancelarMaquina**": Para la ejecución de la máquina y la elimina.

- "**pararMaquina**": Para la máquina, pero no la elimina para poder retomarla.
- "**ejecutarMaquina**": Reanuda la ejecución de la máquina en caso de estar parada.
- "**requisitosCorrectos**" y "**maquinaCorriendo**": Comprueban que Docker este instalado y funcionando y si hay ya una maquina corriendo con el nombre de "altair".
- "**introduccirFlag**" y "**mostrarObjetivos**": Se encargan de administrar las flags.
- "**crearVariasMaquinas**": Simplemente llama a la función encargada de crear una maquina el número de veces introducido.

```
Si se ejecuta el comando sin ningún argumento generará una maquina en el puerto 80, siempre y cuando no esté ya creada. El programa tiene los siguientes argumentos:
- "-p, --port PUERTO": Especifica el puerto, por defecto 80.
- "-f, --flag FLAG": Sirve para introducir la flag y comprobar que se ha obtenido correctamente, si no se especifica FLAG las muestra.
- "-c, --cancel ": Elimina la máquina. Sirve principalmente para rendirse, ya que la maquina se elimina automáticamente cuando obtienes las flags.
- "-s, --stop": Para la máquina, pero no la elimina, puedes continuarla.
- "-r, --run": Ejecuta la maquina en el caso de estar parada.
- "-m, --multi CANTIDAD": Es la segunda funcionalidad del programa, crea el número especificado de máquinas y las almacena en la carpeta "Dockers".
```

Imagen 7

4.4. Modulo "herramientas"

En este módulo se ha ido recopilando funciones necesarias para la ejecución del programa pero que tienen varios usos distintos a lo largo del proyecto. Entre ellas las funciones principales son ejecutar comandos, modificar archivos de texto, generar las flags, obtener valores aleatorios…

Este módulo es usado por todos los otros en algún momento.

4.5. Modulo "maquina"

Este es probablemente el módulo más importante del programa, tiene 2 funciones principales, una crea la máquina ("**crearNuevaMaquina**") y la otra la inicia en caso de haber creado la herramienta en modo "múltiples máquinas" ("**iniciarNuevaMaquina**"). En la siguiente Imagen podemos ver un diagrama de las funciones (*Imagen 8*).

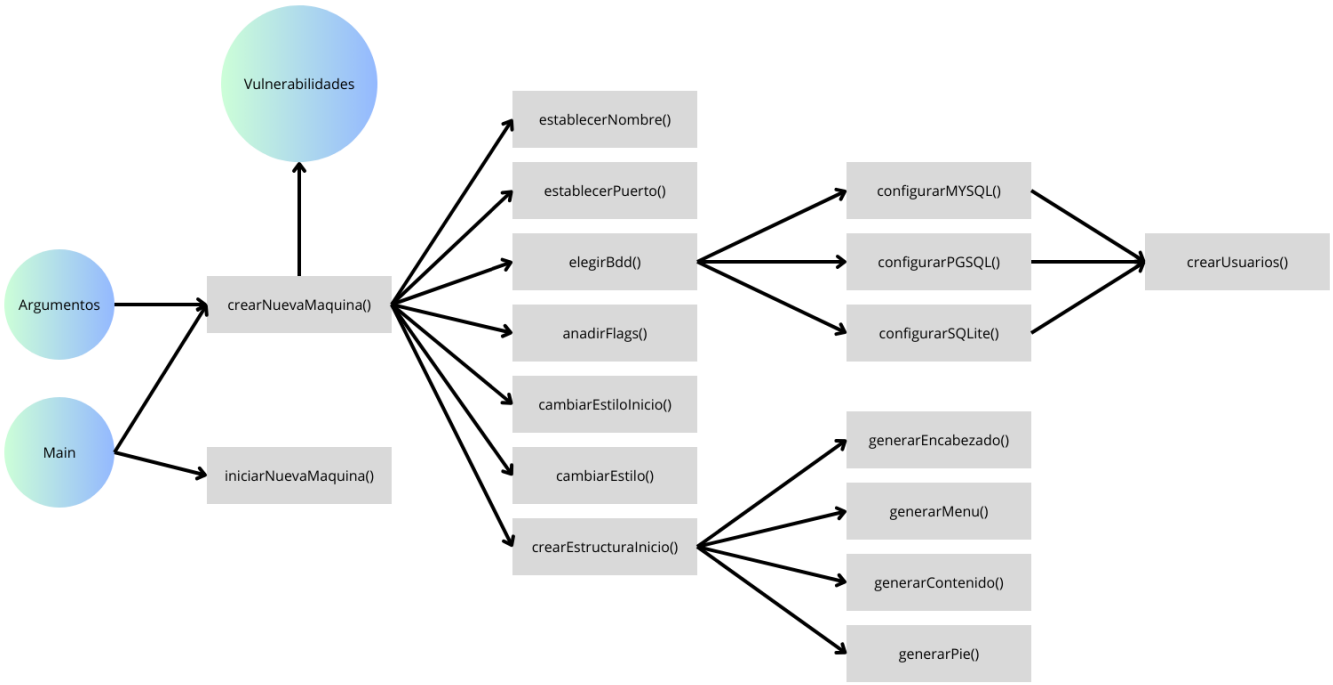


Imagen 8

Para añadir nuevas características a las máquinas es importante entender como funciona la función "**crearNuevaMaquina**". Esta función copia el contenido del ejemplo Docker y llama a las distintas funciones que se van a encargar de modificarlo con los valores necesarios.

Las funciones a las que llama "**crearNuevaMaquina**". Vamos a dividirlas en destinos bloques según su relevancia a la hora de crear características.

Por un lado, tenemos las funciones que sirven para cambiar el estilo de la máquina:

- "**cambiarEstilo**": Eligen un estilo aleatorio de los css que hay en la carpeta assets.
- "**crearEstructuraInicio**": Crea la estructura de la página de forma aleatoria.
- "**cambiarEstiloInicio**": Cambia el estilo de la página con css aleatorios.

Por otro lado, las que añaden y modifican las bases de datos son:

- "**elegirBdd**": Se encarga de elegir, mediante un valor aleatorio que motor de base de datos usar, para eso, llama a las distintas funciones de creación de esa base de datos.
- "**crearUsuarios**": Esta función añade usuarios de ejemplo aleatorio, pero cambia según que base de datos estes usando, por eso es llamada desde la función que crea la base de datos.

El resto de las funciones que debemos tener en cuenta para crear características son las que crean las vulnerabilidades, que se explicarán más adelante.

Las funciones que quedan servirán para la correcta creación de la máquina y no se deberían modificar:

- "**establecerPuerto**": que cambian el puerto que especifiquemos, si lo especificamos.
- "**establecerNombre**": Otorga un nombre, "altair" en caso de ejecutar el programa sin "-m".
- "**anadirFlags**": Introduce las flags dentro de la máquina.

En este módulo también se encuentra la función "**iniciarNuevaMaquina**" que la pondrá en marcha en caso de haber ejecutado el programa sin el argumento "-m".

4.6. Modulo "vulnerabilidades"

Este módulo tiene 3 funciones principales, "**crearVulnerabilidadLogin**", "**crearVulnerabilidadElevacion**" y "**crearVulnerabilidadEjecucion**" cada una va a elegir aleatoriamente un tipo de vulnerabilidad y llamara a la función correspondiente que configure esa vulnerabilidad. esta es la parte más ampliable del proyecto, aquí se pueden ir sumando funciones con vulnerabilidades distintas y luego añadirla a una de estas tres funciones mencionadas. No se va a explicar cada una de estas funciones ya que se explicarán las vulnerabilidades en el apartado de "Vulnerabilidades".

De forma adicional contiene 2 funciones más "**nuevaLineaEnWriteUp**" y "**nuevaVulnEnWriteUp**" que se encargaran de añadir contenido al documento de "**write-up.txt**". Cada vulnerabilidad, al ser creada llamara a estas funciones, y añadirá el documento especifico de esa vulnerabilidad que se encuentra en "doc/vulns", cada uno marcado con un nombre y con una letra ("l", "e", "p" que corresponden a login, ejecución y privilegios respectivamente) para que sean más fáciles de identificar.

5. Integrar nuevas características

En este apartado, se explica el proceso para añadir una nueva vulnerabilidad o sistema al generador de CTFs, de manera que pueda ser seleccionado de forma aleatoria durante la generación de los desafíos. Esta capacidad de integración es crucial para mantener el sistema actualizado y relevante, permitiendo la inclusión continua de nuevos escenarios de seguridad.

5.1. Nueva vulnerabilidad

La incorporación de nuevas vulnerabilidades al generador de CTFs es un proceso sencillo que requiere un entendimiento básico de la estructura y programación de la aplicación, conocimientos que ya se han discutido en la sección anterior de este documento. El sistema facilita la integración de estas nuevas vulnerabilidades, asegurando que el proceso sea lo más eficiente y libre de errores posible. Los pasos para añadir una nueva vulnerabilidad son los siguientes:

1. **Almacenamiento de Archivos Necesarios:**

   El primer paso es organizar y almacenar todos los archivos que se utilizarán para simular la nueva vulnerabilidad. Estos archivos pueden incluir scripts de shell (.sh), archivos comprimidos (.zip), páginas web (.html), entre otros. Todos estos recursos deben ser guardados en la carpeta assets/otros, que actúa como un repositorio central para todos los elementos adicionales necesarios para cada vulnerabilidad.

   Este enfoque garantiza que todos los recursos asociados con la vulnerabilidad estén accesibles y organizados en un único lugar, lo que facilita su manejo y reutilización en el futuro.

2. **Creación del Documento de Resolución (Write-up):**

   Para cada vulnerabilidad que se añada, es importante documentar cómo se puede resolver. Este documento se debe crear en la carpeta doc/vulns y debe tener un formato de texto plano (.txt). La nomenclatura para el nombre de este documento debe seguir el siguiente esquema:

   a. Letra Inicial: Usar una letra que indique el tipo de vulnerabilidad:

      - "l" para vulnerabilidades de login.

- "e" para vulnerabilidades de ejecución.
- "p" para vulnerabilidades de escalada de privilegios.

b. Guion de Separación: Un guion (-) que separa el tipo de vulnerabilidad del nombre descriptivo.

c. Nombre Descriptivo: Un nombre significativo que describa brevemente la vulnerabilidad, por ejemplo, "SQLI" para una inyección SQL.

Ejemplo de nombre de archivo: e-SQLI.txt. Este documento servirá como una guía para los usuarios sobre cómo detectar y mitigar la vulnerabilidad, lo que es esencial para el aprendizaje y la resolución de los desafíos del CTF.

3. Modificación del Archivo `vulnerabilidades.c`:

En la carpeta src, se encuentra el archivo vulnerabilidades.c, que es el corazón del sistema de generación de vulnerabilidades. Aquí es donde se realiza la integración lógica de la nueva vulnerabilidad.

a. Actualización del Módulo de Vulnerabilidad: Dependiendo del tipo de vulnerabilidad (login, ejecución o escalada de privilegios), se debe editar la función correspondiente:

- **crearVulnerabilidadLogin** para vulnerabilidades de login.
- **crearVulnerabilidadEjecucion** para vulnerabilidades de ejecución.
- **crearVulnerabilidadElevacion** para vulnerabilidades de escalada de privilegios.

Dentro de esta función, se debe modificar la variable “tipoVuln” que determina el tipo de vulnerabilidad a crear. Por ejemplo, si la función originalmente contenía “int tipoVuln = rand() % 5;”, deberás aumentar el número en el módulo para incluir la nueva vulnerabilidad, quedando algo como “int tipoVuln = rand() % 6;”.

b. Adición de una Nueva Condición: Dentro de la función modificada, se debe añadir un nuevo bloque else para manejar la nueva vulnerabilidad. Este bloque contendrá dos líneas clave:

Registro en el Write-up:

```
nuevaVulnEnWriteUp("<nombreDoc>", nombreMaquina);
```

Aquí, <nombreDoc> es el nombre del documento de write-up que creaste previamente, y nombreMaquina es la variable que indica la máquina objetivo en el CTF.

Llamada a la Función de Configuración:

return <nuevaFuncion>(nombreMaquina);

Donde <nuevaFuncion> es la función que has creado para configurar y desplegar la vulnerabilidad en la máquina.

4. **Creación de la Función de Configuración de la Vulnerabilidad:**

La última fase consiste en la creación de una función específica dentro de vulnerabilidades.c que será responsable de configurar la nueva vulnerabilidad. Esta función debe manejar todos los aspectos necesarios para que la vulnerabilidad esté presente en el entorno simulado. Esto podría incluir:

- Copiar los archivos necesarios a las ubicaciones correctas en la máquina objetivo.
- Modificar archivos de configuración para activar o desactivar servicios.
- Añadir o editar líneas en archivos clave para simular configuraciones incorrectas o inseguras.

Este es un ejemplo de función (*Imagen 9*):

```
bool rfi(const char *nombreMaquina){
    char rutaInicio[100];
    char copiarFiles[100];
    char copiarFiles2[100];
    char rutaWebpdfsHTML[100];
    char rutaDockerfile[100];

    sprintf(copiarFiles,"cp %srfi.php %s%s/src",rutaOtros,rutaDockers,nombreMaquina);
    sprintf(copiarFiles2,"cp %sphp.ini %s%s/src",rutaOtros,rutaDockers,nombreMaquina);
    sprintf(rutaInicio, "%s%s/src/webContent/inicio.php",rutaDockers, nombreMaquina);
    sprintf(rutaWebpdfsHTML, "%s/rfi.html",rutaOtros);
    sprintf(rutaDockerfile, "%s%s/Dockerfile",rutaDockers, nombreMaquina);

    if(!ejecutarComando(copiarFiles)){
        printf("ERROR: no se ha podido generar la base de datos");
        return false;
    }

    if(!ejecutarComando(copiarFiles2)){
        printf("ERROR: no se ha podido generar la base de datos");
        return false;
    }

    if(!modificarLinea(rutaDockerfile,"##ejecucion##","COPY ./src/rfi.php /var/www/html")){
        printf("ERROR: no se ha podido generar la vulnerabilidad de ejecucion.");
        return false;
    }

    if(!modificarLinea(rutaDockerfile,"##ejecucion2##","COPY ./src/php.ini /etc/php/8.2/apache2/php.ini")){
        printf("ERROR: no se ha podido generar la vulnerabilidad de ejecucion.");
        return false;
    }

    if(!modificarLineaFichero(rutaInicio,"<!--enlace-->",rutaWebpdfsHTML)){
        printf("ERROR: no se ha podido generar la vulnerabilidad de ejecucion.");
        return false;
    }
    return true;
}
```

Imagen 9

5.2. Nueva base de datos

La integración de una nueva base de datos en el generador de CTFs aleatorios requiere un entendimiento profundo de cómo instalar y configurar esa base de datos en particular, más que una comprensión exhaustiva de la programación interna del generador. Los pasos para llevar a cabo esta integración se detallan a continuación:

1. **Creación de una Nueva Carpeta en bbddsConf:**

   El primer paso consiste en organizar los archivos necesarios para la nueva base de datos. Dentro del directorio "bbddsConf", se debe crear una nueva carpeta denominada "bbdd$", donde $ representa el siguiente número disponible en la secuencia (por ejemplo, bbdd1, bbdd2, etc.). Esta convención numérica garantiza un orden lógico y facilita la identificación de las bases de datos configuradas en el sistema.

2. **Creación de los Archivos Necesarios:**

   Dentro de la carpeta recién creada (bbdd$), es necesario crear dos archivos esenciales para la instalación y operación de la base de datos:

   a. **inicio.sh**: Este script se encargará de iniciar el servicio de la base de datos cuando la máquina arranque. Al final del script, es importante incluir las siguientes líneas para asegurar que el servicio web Apache se mantenga en ejecución en primer plano, también es importante el comentario para otras funcionalidades:

      ```
      ##ejecucion##
      /usr/sbin/apachectl -D FOREGROUND
      ```

      Estas líneas aseguran que el servidor web asociado con la base de datos (si es necesario) permanezca activo, permitiendo la interacción continua con la base de datos durante el desafío.

   b. **install.sh**: Este script es responsable de instalar la base de datos en el sistema y de cargar los datos iniciales necesarios para su funcionamiento. Además de realizar la instalación, una función de la app modificará este archivo posteriormente para añadir o ajustar los datos específicos que la base de datos debe contener. Las líneas iniciales del script deben incluir los siguientes comentarios que indican acciones importantes relacionadas con la elevación de privilegios y la ejecución de comandos:

      ```
      ##elevacion##
      ##ejecucion##
      ```

Estos comentarios actúan como marcadores en el script, donde se realizarán operaciones críticas durante la instalación y configuración.

3. Modificación del Archivo maquina.c

El siguiente paso consiste en editar el archivo src/maquina.c para integrar la nueva base de datos en el flujo de selección aleatoria del generador de CTFs. En particular, se debe actualizar la función elegirbdd para tener en cuenta la nueva base de datos:

a. **Ajuste de la Variable numdb:** Incrementa en 1 el módulo de la variable "numdb" para reflejar la inclusión de la nueva base de datos en el conjunto de opciones. Por ejemplo, si originalmente la variable estaba definida como "int numbdd = (rand() % 3) + 1;", ahora debería modificarse a "int numbdd = (rand() % 4) + 1;" (*Imagen 10*).

b. **Adición de un Nuevo else:** A continuación, se debe añadir un nuevo bloque else en la función elegirbdd. Este bloque llamará a la nueva función que se encargará de configurar y desplegar la base de datos en el entorno de la máquina virtual. La función específica que se llama en este else debe corresponder al nombre de la base de datos que se está añadiendo (*Imagen 10*).

```
bool elegirBdd(const char *nombreMaquina){
    int numbdd = (rand() % 3) + 1;
    char copiarBDD[100];
    char rutalogin[100];
    char rutainstall[100];


    sprintf(copiarBDD,"cp -r %sbdd%d %s%s/src/scripts",rutaBdds,numbdd,rutaDockers,nombreMaquina);
    sprintf(rutalogin, "%s%s/src/webContent/login.php",rutaDockers, nombreMaquina);
    sprintf(rutainstall, "%s%s/src/scripts/install.sh",rutaDockers, nombreMaquina);

    if(!ejecutarComando(copiarBDD)){
        printf("ERROR: no se ha podido generar la base de datos");
        return false;
    }
    if (numbdd == 1){
        return configurarMYSQL(rutalogin,rutainstall);
    }else if (numbdd == 2) {
        return configurarPGSQL(rutalogin,rutainstall);
    } else {
        return configurarSQLite(rutalogin,rutainstall);
    }
}
// Fin bases de datos
```

Imagen 10

4. Creación de la Función de Configuración de la Base de Datos:

Finalmente, es necesario crear una nueva función dentro de maquina.c que se encargue de realizar todas las modificaciones necesarias en la configuración de la máquina para asegurar el correcto funcionamiento de la nueva base de datos. Esta función debe realizar varias tareas (*Imagen 11*), incluyendo:

- **Modificación de Archivos de Configuración**: Editar los archivos de configuración del sistema para garantizar que la base de datos funcione correctamente en el entorno del CTF. Esto podría incluir ajustes en las configuraciones de red, seguridad, o servicio.

- **Creación de Usuarios**: Invocar una función para crear usuarios en la base de datos, utilizando los comandos específicos del motor de base de datos que se está integrando.

```
bool configurarMYSQL(const char *rutalogin,const char *rutainstall){
    if(!modificarLinea(rutalogin,"##motorbdd##","\"mysql:host=$host;dbname=$dbname\", $usuario_bd, $contrasena_bd")){
        return false;
    }
    return crearUsuarios(rutainstall,"VARCHAR(255)","INT AUTO_INCREMENT PRIMARY KEY");
}
```

Imagen 11

5.3. Adición y Modificación de Estilos

Este apartado detalla cómo añadir y modificar estilos visuales para personalizar la interfaz del generador de CTFs. Se explicará cómo cambiar el estilo del login, el estilo de inicio, y cómo ajustar el contenido visual de las páginas.

Cambio del Estilo de Login

Para personalizar la apariencia de la página de login, se pueden añadir nuevos estilos CSS que se adapten al HTML existente. El proceso es el siguiente:

1. **Creación del Archivo CSS**: Se debe crear un nuevo archivo de estilo CSS denominado style$.css, donde $ representa el siguiente número disponible en la secuencia (por ejemplo, style1.css, style2.css, etc.). Este archivo debe contener todas las reglas de estilo necesarias para dar un aspecto personalizado a la página de login.

2. **Ubicación del Archivo CSS**: Una vez creado, el archivo CSS debe ser colocado en la carpeta ejemploDocker/src/webcontent/styles. Esta es la ubicación donde se almacenan todos los recursos relacionados con el frontend del generador.

3. **Modificación de la Función cambiarEstilo**: Para que el nuevo estilo de login sea utilizado aleatoriamente por el sistema, es necesario ajustar la función cambiarEstilo en el código. Dentro de esta función, se debe incrementar en 1 el módulo de la variable que selecciona el estilo. Por ejemplo, si originalmente el

código es: sprintf(numeroestilo, "%d", (rand() % 4) + 1); Debería ser modificado a: sprintf(numeroestilo, "%d", (rand() % 5) + 1);.

Esto permitirá que el sistema seleccione de manera aleatoria entre los estilos disponibles, incluido el nuevo estilo que acabas de añadir.

Cambio del Estilo de Inicio

El proceso para cambiar el estilo de la página de inicio es similar al descrito para el login, con algunas variaciones específicas:

1. **Creación del Archivo CSS**: Al igual que con el login, se debe crear un archivo CSS denominado inicio$.css, donde $ es el siguiente número disponible. Este archivo debe definir el estilo visual específico para la página de inicio.

2. **Ubicación del Archivo CSS**: El archivo inicio$.css debe ser guardado en la misma carpeta utilizada para el estilo de login, es decir, en ejemploDocker/src/webcontent/styles.

3. **Modificación de la Función cambiarEstilo**: Asegúrate de que la función que gestiona el estilo de la página de inicio también sea capaz de seleccionar el nuevo estilo de forma aleatoria. Si el código que selecciona el estilo de inicio sigue una lógica similar a la del login, se deberá ajustar de manera análoga para incluir el nuevo archivo CSS.

Modificación del Contenido de la Página de Inicio

Además de los estilos, es posible personalizar el contenido visible en la página de inicio. Esto incluye las imágenes, el texto del menú, el pie de página, y otros elementos que conforman la interfaz del usuario.

Edición del Contenido: Para cambiar estos elementos, es necesario modificar directamente las funciones pertinentes dentro del archivo maquina.c. Cada sección de la página (por ejemplo, las imágenes, el texto del menú, el pie de página) tiene funciones dedicadas en este archivo que se encargan de su renderización y contenido.

Personalización del Texto e Imágenes: Se pueden cambiar las imágenes mostradas, el texto que aparece en el menú, los mensajes del pie de página, y otros textos a través de la edición de estas funciones. Es importante mantener la consistencia visual y temática cuando se realicen estos cambios para asegurar una experiencia de usuario coherente.

6. Vulnerabilidades

El entorno generado presenta una variedad de vulnerabilidades, estratégicamente distribuidas en tres categorías distintas. Estas categorías abarcan vulnerabilidades que pueden manifestarse en la ventana de inicio de sesión, aquellas relacionadas con la ejecución de código dentro del cuerpo de la página web y, por último, aquellas que involucran la escalada de privilegios. diferentes vulnerabilidades que se pueden ejecutar en esta máquina se dividen en tres categorías. Pueden ser vulnerabilidades en la ventana de login, vulnerabilidades de ejecución de código dentro del cuerpo de la web y las vulnerabilidades de la escalada de privilegios.

6.1. Vulnerabilidades login

En esta categoría, se exploran las posibles debilidades en la autenticación y autorización, permitiendo a los participantes identificar y explotar vulnerabilidades relacionadas con la ventana de inicio de sesión. Estas pueden incluir desde técnicas de fuerza bruta hasta fallos en la gestión de sesiones, ofreciendo un espectro diverso de desafíos en el ámbito de la seguridad web.

Usuario y contraseña relajados

La vulnerabilidad de "Usuario y Contraseña Relajados" representa un riesgo significativo en la seguridad de la aplicación web. En este escenario, se ha hecho un uso indebido creando una cuenta de usuario débil o fácilmente adivinable. El fallo también puede recaer en la implantación al permitir la aceptación de estas credenciales. Esta debilidad abre la puerta a ataques de fuerza bruta y compromete la integridad del sistema, exponiendo la información sensible y facilitando el acceso no autorizado.

Explotación del ejemplo implementado en este proyecto

Para este caso se ha utilizado las credenciales de las listas “mysql-betterdefaultpasslist.txt” y “postgres-betterdefaultpasslist.txt” de “**Seclist**” en la ubicación “SecLists/Passwords/Default-Credentials”. También se puede encontrar en el proyecto en “assets/listas/usuarioyPass.txt”.

Con un simple ataque de diccionario con alguna herramienta como “Burpsuite” y estas listas se consigue acceder fácilmente.

Medidas para corregir esta vulnerabilidad

Implementación de Políticas de Contraseñas Fuertes: Establecer y hacer cumplir políticas de contraseñas robustas que requieran combinaciones de caracteres alfanuméricos, mayúsculas, minúsculas y caracteres especiales. Esto dificultará significativamente los ataques de fuerza bruta basados en diccionario.

Aplicar Mecanismos de Bloqueo Automático: Configurar la aplicación para bloquear automáticamente las cuentas después de un número predefinido de intentos fallidos. Esto protegerá contra ataques de fuerza bruta al limitar la cantidad de intentos permitidos en un período de tiempo específico.

Integración de Captchas en Formularios de Inicio de Sesión: Incorporar mecanismos de captcha en los formularios de inicio de sesión para dificultar la automatización de ataques. La resolución de captchas añade una capa adicional de seguridad, dificultando la entrada automatizada.

Establecer Herramientas de Monitoreo de Seguridad: Implementar soluciones de monitoreo continuo que alerten sobre patrones inusuales de actividad de inicio de sesión. Esto permitirá una respuesta proactiva ante posibles intentos de compromiso de cuentas.

Revisión y Actualización de Credenciales Predeterminadas: Regularmente revisar y actualizar las credenciales predeterminadas, eliminando aquellas que sean débiles o fácilmente adivinables. La utilización de listas actualizadas y seguras de contraseñas contribuirá a reforzar la seguridad del sistema.

Campañas de Concientización sobre Seguridad: Llevar a cabo campañas educativas para usuarios finales, enfocadas en la importancia de utilizar contraseñas y prácticas seguras de autenticación. Fomentar la conciencia sobre los riesgos asociados con credenciales débiles y la responsabilidad compartida en la seguridad.

SQL Injection

La amenaza de "SQL Injection" se manifiesta cuando la aplicación web no valida adecuadamente las entradas del usuario, permitiendo la ejecución de comandos SQL no autorizados. Esta vulnerabilidad puede tener consecuencias devastadoras, desde la manipulación de datos almacenados en la base de datos hasta la potencial destrucción de información crítica. Los atacantes pueden aprovechar esta debilidad

para extraer datos confidenciales, modificar registros o incluso comprometer todo el sistema subyacente.

En el panorama actual de seguridad informática, la amenaza de SQL Injection es una de las vulnerabilidades más comunes y explotadas. La dominancia de esta técnica consiste en su simplicidad de ejecución y en la persistente falta de conciencia y prácticas seguras en el desarrollo de aplicaciones web.

Los ataques de inyección SQL están entre las tácticas favoritas de los actores malintencionados, ya que ofrecen una vía directa para el acceso no autorizado y la manipulación de datos. Además, su facilidad de implementación hace que incluso los atacantes menos sofisticados puedan perpetrar con éxito exploits de inyección SQL. Los expertos de ciberseguridad reconocen la importancia de combatir activamente esta amenaza, enfocándose en medidas proactivas, educación continua y la implementación de buenas prácticas de desarrollo seguro.

Explotación del ejemplo implementado en este proyecto

Este, es un SQL injection sencillo, la consulta de SQL del login está programada sin “prepared statment” y el formulario no impide la introducción de caracteres extraños.

Por lo que podemos introducir texto en el apartado nombre del Loguin que nos permita “escapar” de la consulta para consultar otra cosa.

Lo más sencillo sería introducir una cadena como esta “**a' or 1=1 --**“ (*Imagen 12*). Con las comillas simples cerramos el condicional del select de la base de datos, y posteriormente con el “or 1=1” conseguimos que la consulta responda que es cierta. Por ultimo los caracteres “—“ comentan el resto del código SQL para que no haya errores.

Ejecutando ese comando con una contraseña aleatoria habremos iniciado sesión en la máquina teniendo acceso al contenido de la web (*Imagen 13*).

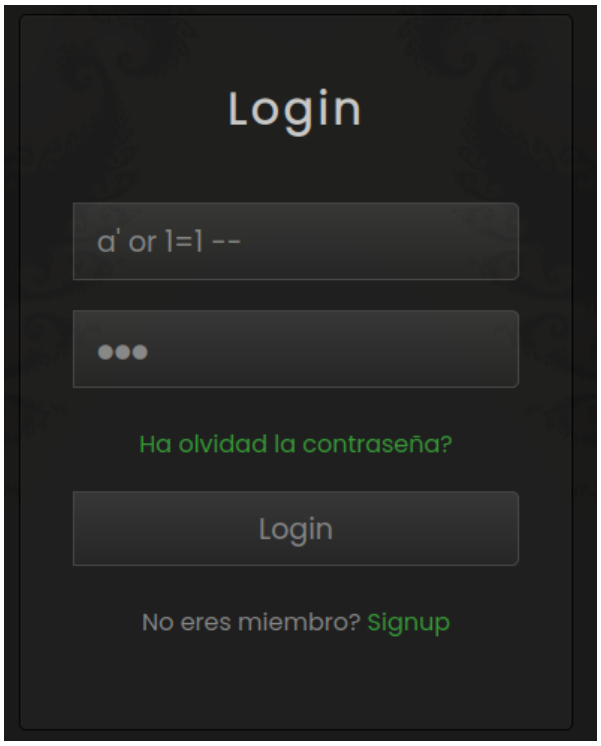


Imagen 12

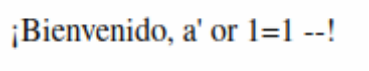


Imagen 13

Medidas para corregir esta vulnerabilidad

Utilización de Consultas Parametrizadas ("Prepared Statements"): Refactorizar el código SQL de la aplicación para emplear "prepared statements". Estas consultas parametrizadas proporcionan una capa de seguridad adicional al separar los datos del usuario de las instrucciones SQL, evitando así la ejecución de comandos no autorizados.

Validación Rigurosa de Datos de Entrada: Implementar una validación estricta de las entradas del usuario para evitar la introducción de caracteres inesperados. Filtrar y sanitizar las entradas para asegurar que solo se ingresen datos válidos, reduciendo así el riesgo de inyecciones SQL.

Adopción de Listas Blancas para Caracteres Permitidos: Establecer listas blancas de caracteres permitidos en las entradas del usuario. Solo permitir caracteres específicos necesarios para las consultas SQL, eliminando cualquier posibilidad de introducción de caracteres maliciosos.

6.2. Vulnerabilidades ejecución de comandos

Dentro del cuerpo de la página web, se pueden encontrar vulnerabilidades que permiten la ejecución de código de manera no autorizada. Estos fallos pueden derivar de inadecuadas validaciones de entrada, escapado insuficiente de caracteres o la explotación de funciones inseguras. La identificación y comprensión de estas vulnerabilidades es esencial para abordar amenazas como la inyección de código y otras formas de ataques avanzados.

Inclusión Remota de Archivos (Remote File Inclusion - RFI)

La vulnerabilidad de Inclusión Remota de Archivos (RFI) es un fallo crítico en aplicaciones web que permite a un atacante incluir archivos remotos en el servidor. Esto puede llevar a la ejecución de código arbitrario, comprometiendo la integridad, confidencialidad y disponibilidad del sistema afectado. Esta vulnerabilidad se manifiesta cuando la aplicación web no valida adecuadamente las entradas del usuario que hacen referencia a archivos, permitiendo que se carguen desde ubicaciones externas (*Imagen 14*).

Imagen 14

Descripción de la Vulnerabilidad

En la máquina afectada, se detecta un formulario que solicita una URL, sugiriendo un posible punto de entrada para una vulnerabilidad RFI. Para verificarlo, se introduce una URL conocida, como la de Google, y se observa que la página redirige a ese sitio a través de la aplicación, confirmando así la presencia de la vulnerabilidad. Este comportamiento indica que la aplicación incluye de manera insegura los archivos remotos sin realizar las validaciones necesarias.

Explotación del ejemplo implementado en este proyecto

La vulnerabilidad RFI puede explotarse de varias maneras, pero en este caso, nos enfocaremos en la forma más sencilla y directa, aprovechando también que la aplicación es vulnerable a Local File Inclusion (LFI).

Crearemos una Shell Reversa en un archivo html. Este archivo permitirá ejecutar comandos en la máquina vulnerable una vez que se incluya remotamente.

A continuación, se inicia un servidor web en la carpeta donde se encuentra el archivo y al mismo tiempo, se pone en escucha un puerto en la máquina local para recibir la conexión de la shell reversa.

Finalmente, se introduce la URL del servidor local en el formulario de la página vulnerable. Esto fuerza a la aplicación a incluir el archivo html, ejecutando el código PHP que lanza la shell reversa. Al hacerlo, la máquina vulnerable contactará con el servidor local, ejecutando el código PHP y estableciendo una conexión de shell remota con el puerto que se había configurado en escucha.

Resumen de la Vulnerabilidad

La vulnerabilidad RFI es extremadamente peligrosa debido a su capacidad para permitir la ejecución remota de código. Este fallo específico en la aplicación se debe a la falta de validación en las entradas del usuario, lo que permite a un atacante incluir archivos desde servidores externos, controlados por él mismo. Al aprovechar esta debilidad, es posible no solo leer archivos locales (LFI), sino también ejecutar comandos en el sistema afectado, comprometiendo gravemente su seguridad.

Cómo Prevenirla:

1. **Validación y Sanitización de Entradas**: Asegurar que todas las entradas de los usuarios que se refieran a archivos sean validadas y sanitizadas adecuadamente. Esto incluye restringir los valores aceptables a rutas de archivos conocidas y seguras, evitando que se pueda acceder a ubicaciones externas.

2. **Configuración Segura del Servidor**: Desactivar funciones de PHP que permitan la inclusión de archivos remotos, como “allow_url_include” y “allow_url_fopen”, cuando no sean necesarias.

3. **Uso de Listas Blancas**: Implementar listas blancas para controlar qué archivos pueden ser incluidos o cargados, restringiendo las opciones a un conjunto controlado de archivos internos.

4. **Monitoreo y Alerta de Actividades Sospechosas**: Configurar sistemas de detección de intrusiones y monitoreo de registros para identificar intentos de explotación de vulnerabilidades como RFI, y responder rápidamente a incidentes de seguridad.

La correcta implementación de estas medidas puede prevenir que vulnerabilidades como RFI sean explotadas, protegiendo la aplicación web y los datos que maneja.

Inyección de Comandos (Command Injection)

La vulnerabilidad de Inyección de Comandos (Command Injection) es una de las más graves en el ámbito de la seguridad web. Esta vulnerabilidad permite a un atacante inyectar y ejecutar comandos del sistema operativo a través de una aplicación web que no valida adecuadamente las entradas del usuario. Como resultado, el atacante puede ejecutar comandos arbitrarios en el servidor objetivo, lo que puede llevar al compromiso total del sistema. Este tipo de vulnerabilidad surge cuando una aplicación utiliza entradas del usuario para construir comandos del sistema sin realizar una validación o saneamiento adecuado de dichos datos (*Imagen 15*).

Imagen 15

Descripción de la Vulnerabilidad

En la máquina afectada, se ha detectado un formulario que solicita un comando a ejecutar. Tras realizar pruebas simples, como introducir cat /etc/passwd, se confirma que la aplicación ejecuta los comandos proporcionados sin ninguna validación o restricción adecuada. Este comportamiento confirma la presencia de una vulnerabilidad RCE.

Explotación del ejemplo implementado en este proyecto

Dada la capacidad de ejecutar comandos directamente en el servidor, uno de los objetivos comunes de un atacante sería obtener una shell reversa para tener acceso interactivo al sistema comprometido. Sin embargo, en este caso particular, existen ciertas restricciones que deben ser abordadas para explotar la vulnerabilidad exitosamente.

Inicialmente, se podría intentar lanzar una shell reversa utilizando bash con la siguiente sintaxis y poner en escucha el puerto en la máquina atacante.

Sin embargo, este método no funcionará en este caso porque el usuario que ejecuta el comando en el servidor (www-data) utiliza la shell /bin/sh, que no es compatible con la sintaxis de bash utilizada para la shell reversa. Esta es una limitación importante que impide el uso de este método directamente.

Dado que el método anterior no es viable, podemos buscar alternativas. Sabemos que php está instalado en el servidor, y podemos utilizarlo para lanzar la shell reversa. Ejecutando una Shell reversa utilizando php y poniendo el servidor atacante en escucha tendríamos acceso interactivo al sistema comprometido bajo los privilegios del usuario www-data.

Resumen de la Vulnerabilidad

La vulnerabilidad RCE detectada en esta aplicación web es extremadamente peligrosa, ya que permite la ejecución de comandos arbitrarios en el servidor remoto. En este caso, la vulnerabilidad fue explotada para obtener una shell reversa utilizando PHP, después de que fallara el intento inicial con bash debido a la incompatibilidad con /bin/sh.

Cómo Prevenirla:

1. **Validación Estricta de Entradas**: Es crucial validar y sanitizar todas las entradas del usuario, especialmente aquellas que se ejecutan en el servidor. Esto incluye bloquear la ejecución de comandos directamente a partir de entradas del usuario o utilizar listas blancas para permitir solo comandos específicos y seguros.

2. **Deshabilitar Funcionalidades Innecesarias**: Si la funcionalidad de ejecutar comandos desde el formulario no es absolutamente necesaria, debería deshabilitarse. En su lugar, se pueden

implementar métodos más seguros para las tareas que requieran comandos del sistema.

3. **Aislamiento del Entorno de Ejecución**: Utilizar entornos de ejecución aislados, como contenedores, para minimizar el impacto de una explotación exitosa. Esto limita la capacidad de un atacante de comprometer el sistema completo si logra explotar la vulnerabilidad.

4. **Monitoreo y Alertas**: Implementar un sistema de monitoreo que detecte y alerte sobre ejecuciones de comandos inusuales o sospechosos. Esto puede ayudar a identificar y responder rápidamente a intentos de explotación.

5. **Auditoría de Código**: Revisar periódicamente el código fuente para identificar posibles vulnerabilidades RCE, prestando especial atención a las funciones que interactúan directamente con el sistema operativo o que permiten la ejecución de comandos.

Al implementar estas medidas, se puede reducir significativamente el riesgo de que una vulnerabilidad RCE sea explotada, protegiendo la integridad y seguridad del sistema.

Transversal de Directorios (Path Traversal) y Subida de Archivos Sin Restricciones

Este equipo presenta una combinación peligrosa de dos vulnerabilidades críticas: Path Traversal (Transversal de Directorios) y Subida de Archivos Sin Restricciones. La explotación de estas vulnerabilidades de manera conjunta puede llevar a la ejecución remota de código y, por ende, a la toma de control del sistema (*Imagen 16*).

Imagen 16

Descripción de las Vulnerabilidades

Path Traversal (Transversal de Directorios) es una vulnerabilidad que permite a un atacante acceder a archivos y directorios fuera del directorio raíz de la aplicación web. Esto se logra manipulando las rutas de los archivos enviados a la aplicación, utilizando secuencias de escape como ../../ para "retroceder" en la jerarquía de directorios del sistema de archivos.

Subida de Archivos Sin Restricciones es otra vulnerabilidad crítica que permite a un atacante subir archivos a un servidor sin restricciones adecuadas. Si los archivos maliciosos subidos pueden ejecutarse en el servidor, el atacante puede comprometer completamente el sistema.

Explotación del ejemplo implementado en este proyecto

En esta máquina, se pueden explotar ambas vulnerabilidades en conjunto para obtener una shell remota.

Primero, creamos un archivo .php que contiene código PHP para ejecutar una shell reversa. Este archivo se subirá al servidor a través de un formulario vulnerable que permite la subida de archivos sin las restricciones adecuadas. Este archivo PHP, una vez ejecutado en el servidor, establecerá una conexión de shell inversa con la máquina atacante.

Utilizando el formulario vulnerable, subimos el archivo index.php al servidor. Dado que no existen restricciones en el formulario, el archivo se cargará y almacenará en una ubicación dentro del servidor. Sin embargo, la aplicación no proporcionará directamente la ubicación completa del archivo subido, lo que es común en estos escenarios.

Para ejecutar el archivo PHP malicioso, utilizaremos la vulnerabilidad de Path Traversal. Esta nos permitirá escapar del directorio predeterminado donde la aplicación busca archivos y acceder directamente al archivo que acabamos de subir. Esto se logra insertando una ruta manipulada en un formulario que espera un archivo.

Poniendo a atacante en escucha y ejecutando el archivo PHP en la victima escapando de la ruta normal con “../../../../../../..” podremos obtener acceso al objetivo.

Resumen de las Vulnerabilidades

La combinación de Path Traversal y Subida de Archivos Sin Restricciones representa una amenaza grave para la seguridad del sistema. La capacidad de subir archivos maliciosos y luego ejecutarlos mediante la manipulación de rutas permite a un atacante comprometer completamente el servidor.

Cómo Prevenirlas:

1. **Validación y Restricción de Rutas**: Implementar un control estricto sobre las rutas de los archivos que la aplicación puede acceder. Utilizar listas blancas para restringir las rutas a ubicaciones específicas y seguras, y validar rigurosamente cualquier entrada del usuario que manipule rutas.

2. **Control de Subida de Archivos**: Imponer restricciones estrictas en la subida de archivos. Esto incluye:
   - Validación del tipo de archivo: Permitir solo ciertos tipos de archivos seguros, como imágenes o documentos, y bloquear la subida de archivos ejecutables.
   - Limitación de ubicaciones: Almacenar los archivos subidos en directorios seguros donde no puedan ser ejecutados directamente.
   - Análisis de contenido: Utilizar herramientas de análisis para detectar contenido malicioso en los archivos subidos antes de permitir su almacenamiento.

3. **Ejecutar Servidores en Entornos Aislados**: Implementar contenedores o máquinas virtuales para ejecutar la aplicación web, minimizando el impacto de cualquier explotación exitosa y limitando el acceso del atacante al sistema operativo subyacente.

4. **Monitoreo y Registro**: Implementar un sistema de monitoreo que registre las actividades relacionadas con la subida de archivos y el acceso a rutas críticas. Esto puede ayudar a detectar intentos de explotación y responder a incidentes de seguridad de manera oportuna.

Con estas medidas, se pueden mitigar los riesgos asociados a estas vulnerabilidades, protegiendo el servidor de posibles ataques y compromisos.

Fallos en la Autenticación y Control de Acceso

La vulnerabilidad identificada en el sistema de autenticación y gestión de usuarios mediante json web tokens presenta dos fallos críticos que pueden comprometer gravemente la seguridad de la aplicación web. En primer lugar, la excesiva revelación de información en la documentación, que expone detalles sensibles, como el nombre del usuario administrador, facilitando potencialmente ataques dirigidos. En segundo lugar, el código subyacente permite a un usuario administrador ejecutar comandos arbitrarios en la máquina remota, exponiendo la aplicación a riesgos de ejecución remota de comandos.

Explotación del ejemplo implementado en este proyecto

Esta vulnerabilidad consta de una página, a la que accedemos mediante virtual hosting (*Imagen 17*). Que describe cómo se puede utilizar los tokens para acceder a la base de datos que contiene (*Imagen 18*).

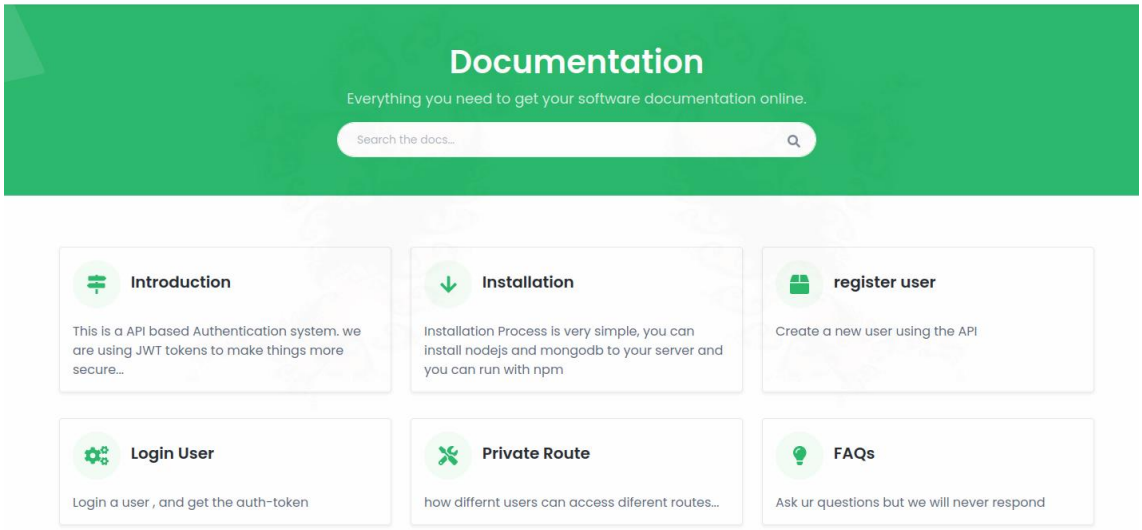


Imagen 17

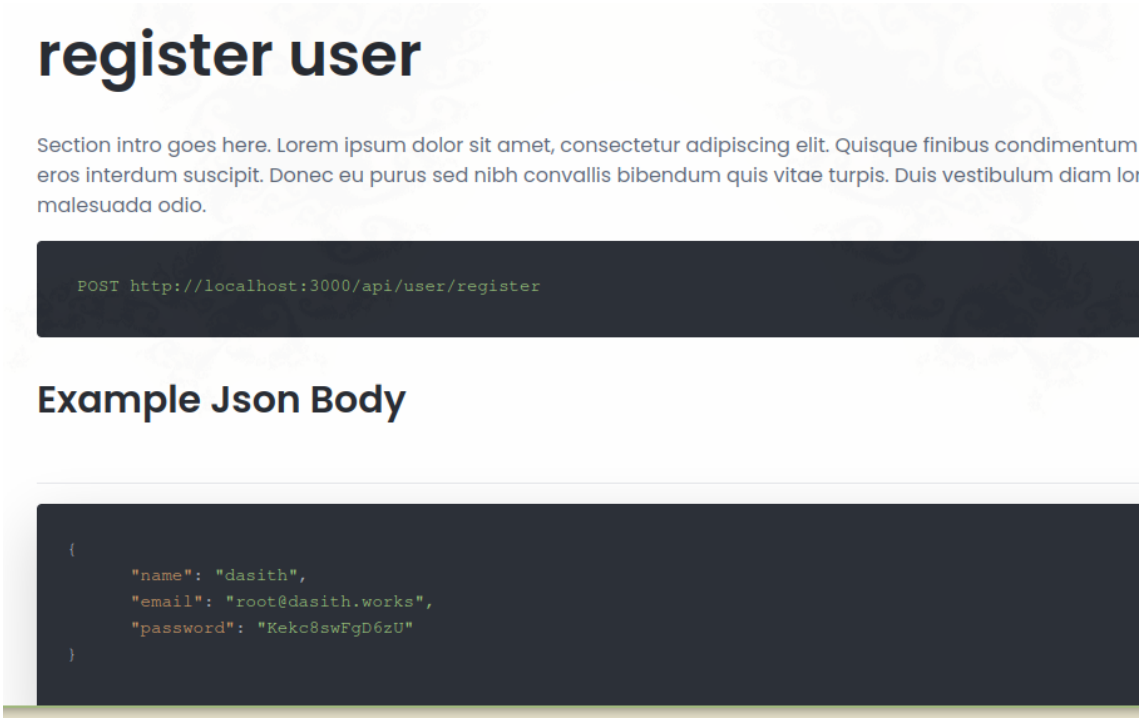


Imagen 18

También tiene un apartado para descargar el código. haciendo fuzzing o leyendo el código podemos ver cómo hay una dirección, "/api/logs", que ejecuta un comando en la maquina (*Imagen 19*) con la única restricción que el nombre de usuario responda a "theadmin".

```
if (name == 'theadmin'){
    const getLogs = `git log --oneline ${file}`;
    exec(getLogs, (err , output) =>{
        if(err){
            res.status(500).send(err);
            return
        }
        res.json(output);
    })
}
```

Imagen 19

Como podemos intuir en la imagen podemos escapar de ese comando introduciendo simplemente ";". Por ejemplo, así: "file=/etc/passwd;whoami". La única pega es que necesitamos ser el usuario "theadmin", dicho usuario no está creado en la máquina, así que podemos crearlo con una simple petición a "/api/user/register".

En resumen, esta vulnerabilidad se debe ante tres grandes errores:

1. Divulgación de Información Sensible: La documentación detallista de la web revela información crítica, como el nombre del usuario administrador. Esto puede ser aprovechado por atacantes para realizar ataques de ingeniería social, ataques dirigidos o intentos de suplantación de identidad, comprometiendo la confidencialidad y seguridad de la aplicación.

2. Ejecución Remota de Comandos: El código mal implementado permite al usuario administrador ejecutar comandos en la máquina remota sin una adecuada validación de seguridad. A través de la manipulación de parámetros, un atacante puede inyectar comandos maliciosos, comprometiendo la integridad y confidencialidad del sistema. Esto podría conducir a la pérdida de datos, daños en la infraestructura o incluso tomar el control total del sistema.

3. Fracaso en la Autenticación del Usuario Administrador: La falta de un proceso robusto de autenticación permite la creación de un usuario administrador sin restricciones, simplemente coincidiendo con el nombre. Esto posibilita a un atacante crear un usuario falso con el nombre adecuado y ejecutar comandos en la máquina remota, aprovechando la debilidad en el control de acceso.

Como resolver estos errores:

Reducir Detalles Sensibles: Actualizar la documentación para limitar la divulgación de detalles sensibles, como nombres de usuarios administradores. Proporcionar información suficiente para el uso adecuado de la API sin exponer información crítica que pueda ser aprovechada por posibles atacantes.

Implementar Validaciones de Acceso: Reforzar el sistema de autenticación para verificar no solo el nombre del usuario, sino también la autenticidad del token y la autorización para realizar acciones específicas. Esto evitará la creación no autorizada de usuarios administradores y garantizará un control de acceso adecuado.

Revisar y Corregir el Código Fuente: Realizar una auditoría exhaustiva del código fuente para identificar y corregir vulnerabilidades. Asegurarse de que las funciones críticas, como la ejecución de comandos, se realicen de manera segura y con una validación adecuada de los parámetros de entrada.

Implementar Pruebas de Seguridad Regularmente: Integrar pruebas de seguridad periódicas en el ciclo de desarrollo para identificar y abordar posibles vulnerabilidades antes de que lleguen a producción. Estas pruebas deben incluir escenarios de ejecución remota de comandos y evaluar la resistencia del sistema a posibles manipulaciones.

Promover Prácticas Seguras en el Desarrollo: Capacitar a los desarrolladores sobre las mejores prácticas de seguridad, haciendo hincapié en la importancia de validar entradas, implementar controles de acceso sólidos y evitar la divulgación innecesaria de información en la documentación.

Herramienta desactualizada (pdfkit)

El sistema utiliza pdfkit para generar archivos PDF a partir de URL proporcionadas a través de un formulario web. La versión 0.8.6 de pdfkit (CVE-2022-25765), utilizada en este contexto, presenta una vulnerabilidad que puede ser aprovechada por un atacante para ejecutar comandos arbitrarios en el sistema que aloja la aplicación.

Explotación del ejemplo implementado en este proyecto

Para detectarla basta con una herramienta como "**exiftool**" que nos permita ver los metadatos del documento descargado (*Imagen 20*).

```
❯ exiftool 8dz1tpqurimn51c2i19qe46xlrebgfr4.pdf
ExifTool Version Number         : 12.70
File Name                       : 8dz1tpqurimn51c2i19qe46xlrebgfr4.pdf
Directory                       : .
File Size                       : 49 kB
File Modification Date/Time     : 2023:12:22 02:25:05+01:00
File Access Date/Time           : 2023:12:22 02:25:06+01:00
File Inode Change Date/Time     : 2023:12:22 02:25:05+01:00
File Permissions                : -r--------
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Page Count                      : 1
Creator                         : Generated by pdfkit v0.8.6
```

Imagen 20

Buscando información sobre esa versión vemos como contiene una vulnerabilidad en la que introduciéndole una URL valida seguido de "**?name=%20``**" y añadiéndole un comando entre los acentos podremos ejecutarlo. Con un comando "sleep" podríamos comprobar que funciona (*imagen 21*). Y podríamos ejecutar una shell reversa directamente.

Imagen 21

La explotación exitosa de esta vulnerabilidad podría permitir a un atacante ejecutar comandos maliciosos con los privilegios del proceso de generación de PDF. Esto podría llevar a consecuencias graves, como la manipulación no autorizada del sistema, pérdida de datos o incluso la toma completa del control del servidor.

La importancia de mantener actualizadas las herramientas y bibliotecas de software, como pdfkit, se destaca en esta vulnerabilidad. Las actualizaciones de software a menudo incluyen correcciones de seguridad que abordan vulnerabilidades conocidas. No mantener el software al día deja al sistema vulnerable a amenazas conocidas y potencialmente explotables.

En este contexto específico, la actualización a una versión más reciente de pdfkit que parchee la vulnerabilidad CVE-2022-25765 es crucial. Los desarrolladores y administradores de sistemas deben adoptar una política proactiva de actualización de software para garantizar la seguridad y resistencia de sus aplicaciones contra amenazas en constante evolución.

Las medidas para evitar este tipo de vulnerabilidades son:

Actualización de Software: Realizar una actualización inmediata de pdfkit a una versión más reciente y segura que haya corregido la vulnerabilidad CVE-2022-25765. Mantener todo el software, incluyendo

el sistema operativo y sus dependencias, actualizado para mitigar futuras vulnerabilidades.

Monitorización y Detección: Implementar sistemas de detección de intrusiones para monitorear y alertar sobre patrones de comportamiento sospechoso o intentos de explotación. Revisar y analizar regularmente los logs del sistema en busca de actividades inusuales o intentos de ejecución de comandos no autorizados.

Principio de Menor Privilegio: Configuración de Permisos: Ajustar los permisos del sistema y de la aplicación para seguir el principio de menor privilegio, limitando los accesos solo a lo necesario para el funcionamiento correcto de la aplicación.

6.3. Vulnerabilidades de escalada de privilegios

La última categoría se enfoca en la escalada de privilegios, un aspecto crucial en la seguridad de sistemas. Aquí, los participantes se enfrentarán a desafíos relacionados con la obtención de privilegios adicionales, explorando las posibles debilidades que podrían permitir la elevación de privilegios desde un nivel de usuario estándar hasta un nivel administrativo.

Archivo con sudo

La vulnerabilidad relacionada con "Archivo con Sudo" se centra en la incorrecta configuración de los privilegios administrativos a través del comando sudo. Cuando un archivo específico tiene permisos para ser ejecutado con privilegios elevados mediante sudo, existe el riesgo de que un atacante pueda manipular este archivo para ejecutar comandos no autorizados con privilegios de administrador. Esta debilidad puede resultar en la comprometida integridad del sistema, permitiendo acciones no deseadas que pudieran afectar la estabilidad y seguridad del entorno.

Explotación del ejemplo implementado en este proyecto

Para descubrir si estamos ante esta vulnerabilidad basta con ejecutar el comando “**sudo -l**”. Si es así nos aparecerá un archivo con permisos de sudo “**ALL ALL=(ALL) NOPASSWD: #archivo#**” siendo “#archivo#” el archivo con permisos de sudo (*Imagen 22*).

El archivo será uno recogido en la lista de "GTFOBins" (https://gtfobins.github.io/) en esa página se detalla como explotar la vulnerabilidad para cada caso.

```
www-data@e0acee17a886:/var/www/html$ sudo -l
sudo -l
Matching Defaults entries for www-data on e0acee17a886:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User www-data may run the following commands on e0acee17a886:
    (ALL) NOPASSWD: /usr/bin/taskset
www-data@e0acee17a886:/var/www/html$
```

Imagen 22

Para mitigar esta amenaza y fortalecer la seguridad del sistema, se proponen las siguientes medidas:

Reevaluación de Configuraciones Sudo: Realizar una revisión exhaustiva de los archivos con permisos sudo existentes. Limitar el uso del comando sudo a los archivos y comandos esenciales, reduciendo la superficie de ataque y minimizando la posibilidad de ejecución de comandos maliciosos.

Adopción de Listas Blancas para Archivos Ejecutables: Establecer listas blancas que enumeren específicamente los archivos ejecutables permitidos mediante sudo. Esto restringirá la ejecución de comandos a aquellos archivos esenciales y predefinidos, mitigando el riesgo de manipulación maliciosa.

Realizar Auditorías Periódicas de Configuración de Sudo: Ejecutar auditorías regulares para evaluar la configuración de sudo en busca de posibles debilidades. Estas auditorías deben incluir la revisión de archivos con privilegios sudo y la identificación de posibles puntos de explotación.

Archivo con SUID

La vulnerabilidad asociada con "Archivo con SUID" se presenta cuando un archivo posee el bit SUID configurado, otorgando al usuario que lo ejecuta temporales privilegios elevados. Esta configuración puede ser explotada por un atacante para realizar acciones que, de otra manera,

estarían fuera de su alcance. Identificar y corregir adecuadamente estas configuraciones es crucial, ya que un mal uso de archivos con SUID podría resultar en la ejecución de comandos con privilegios de usuario "root", comprometiendo así la seguridad del sistema.

Explotación del ejemplo implementado en este proyecto

Esta vulnerabilidad es explotada si tenemos configurado un archivo del sistema con SUID, para encontrar este archivo basta con ejecutar el comando "**find / -perm /6000 2>/dev/null**" (*Imagen 23*). No todos los archivos que aparecen aquí son vulnerables, habrá que buscar cual está en la lista de "GTFOBins" (https://gtfobins.github.io/).

Para explotarlo habrá que seguir los datos de la página de "GTFOBins" para cada caso.

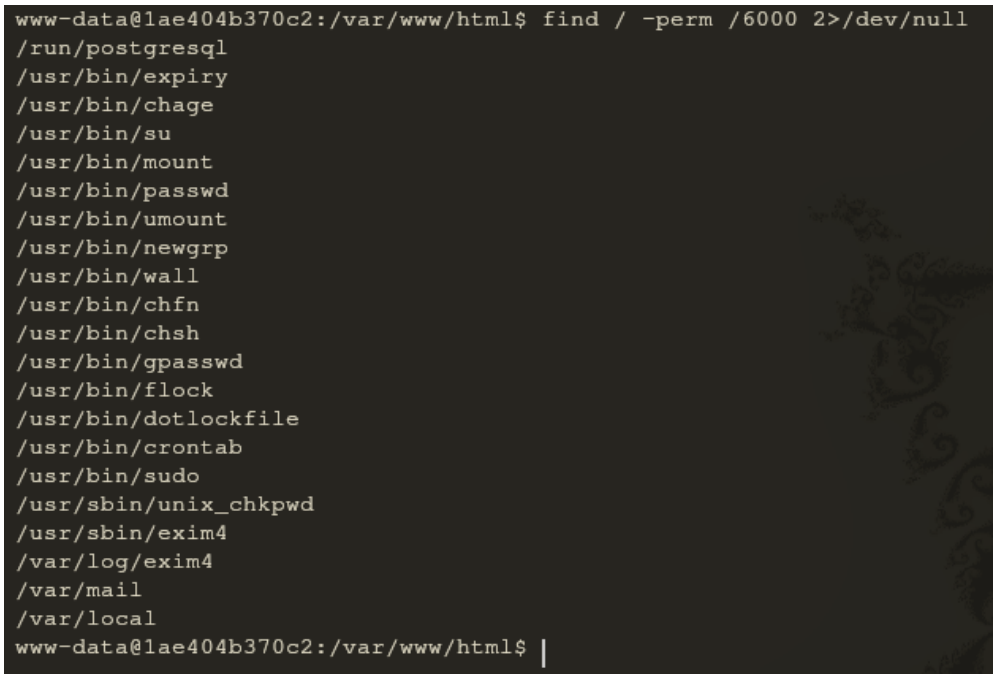


Imagen 23

La mitigación de esta amenaza es muy parecida a la de "SUDO", aunque de todos modos va a ser recalcada a continuación:

Reevaluación de Configuraciones SUID: Realizar una revisión exhaustiva de los archivos que tienen configurado el bit SUID. Limitar el uso de SUID solo a archivos esenciales y necesarios para el funcionamiento del sistema, reduciendo así la superficie de ataque.

Identificación y Desactivación de SUID en Archivos No Esenciales: Identificar archivos que no requieren SUID y desactivar el bit SUID en

aquellos que no son esenciales. Esto minimizará las oportunidades para la explotación de archivos con SUID innecesarios.

Realizar Auditorías Regulares de Configuración: Ejecutar auditorías periódicas para evaluar la configuración del bit SUID en busca de posibles debilidades. Identificar y corregir configuraciones incorrectas o innecesarias para fortalecer la seguridad del sistema.

7. CONCLUSIONES

El desarrollo de este proyecto ha resultado en la creación de un entorno funcional y versátil para la práctica de ciberseguridad, centrado en la simulación de máquinas web vulnerables. A través del uso de Docker, se ha logrado diseñar un sistema que permite generar entornos variados con vulnerabilidades específicas, ofreciendo a los usuarios una plataforma para explorar y mejorar sus habilidades en hacking ético y pruebas de penetración.

Capacidades del Proyecto: La herramienta desarrollada permite a los usuarios crear y gestionar múltiples máquinas con vulnerabilidades aleatorias, lo que simula las condiciones impredecibles y diversas de los entornos reales. Aunque no se han realizado pruebas extensivas para evaluar su efectividad educativa, la estructura del proyecto facilita su uso en el aprendizaje autónomo y en la preparación para retos de ciberseguridad.

Flexibilidad y Expansión: Una de las fortalezas del proyecto es su capacidad para integrar nuevas características y vulnerabilidades sin necesidad de realizar modificaciones significativas en el código base. Esta flexibilidad asegura que la herramienta pueda adaptarse y evolucionar a medida que cambian las necesidades en el campo de la ciberseguridad.

Enfoque en Vulnerabilidades Críticas: El proyecto se centra principalmente en la explotación de vulnerabilidades relacionadas con la ejecución remota y la elevación de privilegios, áreas críticas en la seguridad web. Este enfoque específico permite a los usuarios familiarizarse con técnicas esenciales en la identificación y mitigación de amenazas.

En resumen, este proyecto proporciona una base sólida para el desarrollo de habilidades en ciberseguridad, ofreciendo un entorno controlado y configurable que puede ser utilizado tanto para la práctica individual como para la creación de escenarios de entrenamiento en plataformas más amplias.

8. REFERENCIAS

GitHub del proyecto:

https://github.com/ibantxu12/SSHKey_openldap

Información sobre las vulnerabilidades:

https://gtfobins.github.io/

Información sobre la tecnología:

https://www.docker.com/

https://docs.docker.com/engine/install/debian/

https://jolthgs.wordpress.com/2019/09/25/create-a-debian-container-in-docker-for-development/

https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-debian-11

https://www.php.net/manual/es/intro.pdo.php

https://www.codingnepalweb.com/free-login-registration-form-html-css/

Noticias Ciberseguridad:

https://www.20minutos.es/tecnologia/ciberseguridad/ciberataque-expone-miles-datos-personales-medicos-guardia-civil-5236785/

https://www.xataka.com/robotica-e-ia/no-todo-color-rosa-mundo-ia-microsoft-alerta-llegada-ciberataques-devastadores

https://www.europapress.es/catalunya/noticia-agencia-ciberseguretat-catalunya-gestiono-mas-5000-millones-ciberataques-2023-20240730173825.html