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, fecha de defensa

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Contenido

1. 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.

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 aprender a 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. En conjunto, estas plataformas desempeñan 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.

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 la capacidad de adaptarse a entornos dinámicos, una habilidad esencial en el campo de la seguridad informática.

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 la explotación de vulnerabilidades en Docker en sí mismo, sino en la utilización de Docker como una herramienta eficaz para la creación de entornos virtualizados.

La arquitectura de contenedores de Docker no solo facilita la replicación de entornos, sino que también añade una capa adicional de aislamiento, contribuyendo así a reforzar la seguridad de las máquinas generadas. Explorar y comprender cómo interactúan las vulnerabilidades en estos entornos virtualizados proporciona una perspectiva práctica y valiosa para aquellos que buscan fortalecer sus habilidades en ciberseguridad, sin comprometer la integridad de la herramienta fundamental utilizada en el proceso de creación.

1.3. La Importancia Crucial de la Seguridad Web

La seguridad web se ha convertido en un aspecto vital en un mundo interconectado. Con la creciente sofisticación de las amenazas cibernéticas, la protección de las aplicaciones web se vuelve esencial. Este proyecto no solo busca proporcionar un entorno educativo para comprender y abordar 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 programar en lenguaje C no solo responde a sus características inherentes de eficiencia y control, sino que se focaliza en el desarrollo del programa que orquesta la ejecución de Docker. Al ser un lenguaje ampliamente enseñado y utilizado en la carrera, su dominio facilita la implementación eficiente de este programa, permitiendo una comprensión profunda de los aspectos de seguridad y vulnerabilidades que puedan surgir en la gestión de las máquinas web. Esta elección estratégica no solo agiliza el proceso de coordinación con Docker, sino que también proporciona una base sólida para abordar desafíos específicos relacionados con la seguridad informática.

En resumen, este proyecto no solo se presenta como una herramienta para aprender sobre la seguridad informática, sino como un medio para fomentar una mentalidad de seguridad desde el inicio del ciclo de desarrollo. Al ofrecer un entorno de aprendizaje práctico y dinámico, se busca preparar a los profesionales para los desafíos en constante evolución del paisaje digital.

2. 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. Estos objetivos se desglosan en los siguientes aspectos:

1.2. Aleatoriedad Integral: Simulando Escenarios del Mundo Real

Este proyecto busca incorporar un elemento de aleatoriedad integral en la generación de máquinas web, incluyendo 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 amenazas en entornos del mundo real, proporcionando a los participantes una experiencia de aprendizaje práctica y adaptativa.

1.3. Vulnerabilidades Web: Ejecución Remota y Elevación de Privilegios

El componente central del proyecto se enfoca en la introducción de vulnerabilidades en las máquinas web simuladas. 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 subyacente.
- **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 reconoce 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 internalizar y aplicar los conocimientos adquiridos.

La disponibilidad de write-ups personalizados no solo facilitará la retroalimentación constructiva, sino que también fomentará la comunidad de aprendices al compartir conocimientos y enfoques diversos. De esta manera, 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.

3. FUNCIONAMIENTO

Esta herramienta se puede usar de dos formas distintas, una está directamente dedicada al alumno, ejecutando el programa en una consola generara una maquina (*Imagen 1*) en el puerto 80 (si no se especifica).

Una vez que se exploten las vulnerabilidades de esa máquina se obtendrán dos "flags" una cuando se obtiene acceso remoto a la maquina y otra cuando se consigue acceso privilegiado. Estas flags se pueden introducir mediante "-f" para completar la maquina (*Imagen 2*).

La otra forma de utilizarla tiene más funciones, mediante "-m" se pueden crear un número ilimitado de máquinas, las cuales irán a la carpeta "dockers" del proyecto (*Imagen 3*). Esas máquinas se pueden utilizar para, por ejemplo, subirlas a una herramienta de CTFs como hackthebox, tryhackmy…

```
❯ ./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

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

Imagen 3

Cada máquina que se cree contendrá, a parte de los archivos necesarios para su creación, dos archivos "start.sh" y "stop.sh" que ejecutan y paran la máquina. Además, tendremos un "write-up.txt" que detallara la su resolución.

4. IMPLEMENTACIÓN

Este proyecto consta de varios archivos necesarios para la creación de la maquina a parte del propio programa escrito en C. En este apartado se explicará la distribución del proyecto y los distintos módulos en C.

La carpeta principal consta del programa compilado, un "Makefile" para facilitar la compilación de este y el "README".

También encontramos la carpeta "assets" la cual más adelante se explicará su función.

La carpeta "doc" que contiene la documentación necesaria del programa. La carpeta "dockers" donde Irán las maquinas una vez creadas.

Y por último las carpetas "src", "include" y "obj", donde irán las ".c", ".h" y ".o" respectivamente, más adelante se explicarán en detalle los distintos módulos del programa. Estos módulos, como se puede ver en la imagen (*Imagen 4*), se relacionan de una forma muy simple, el "Main" llama al módulo "Argumentos" según que argumento le pasemos o crea una maquina llamando al modulo "Maquina". "Argumentos" crea también maquinas llamando a "Maquina". "Maquina" usa el módulo de "Vulnerabilidades" para añadirlas. Y, por último, estos tres se apoyan del modulo herramientas, que recoge una serie de funciones valiosas no específicas.

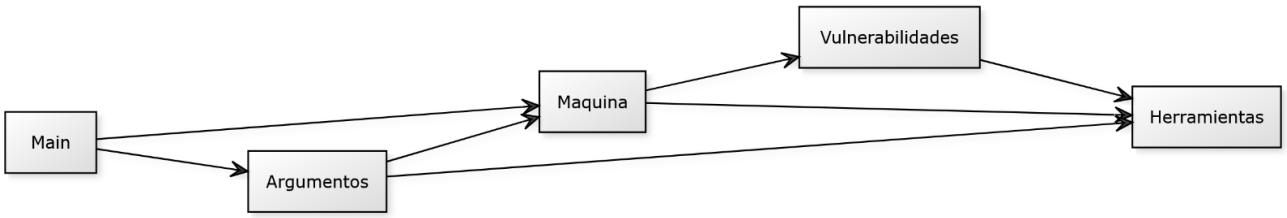

Imagen 4

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

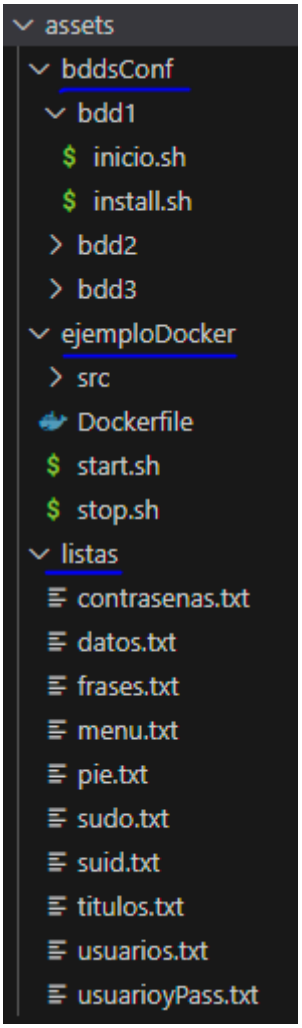

Imagen 5

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.

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.

Tenemos la función "**mostrarAyuda**" que saca por pantalla la información sobre los argumentos. "**cancelarMaquina**", "**pararMaquina**" y "**ejecutarMaquina**" que eliminan la máquina, la pausan y la reanudan respectivamente ("-c","-s","-r"). Las funciones "**requisitosCorrectos**" y "**maquinaCorriendo**" que comprueban que

Docker este instalado y funcionando y si hay ya una maquina corriendo con el nombre de “altair”. También dispone de las funciones “**introduccirFlag**” y “**mostrarObjetivos**” para administrar las flags. Y, por último, “**crearVariasMaquinas**” que llama a la función encargada de crear una maquina el número de veces introducido.

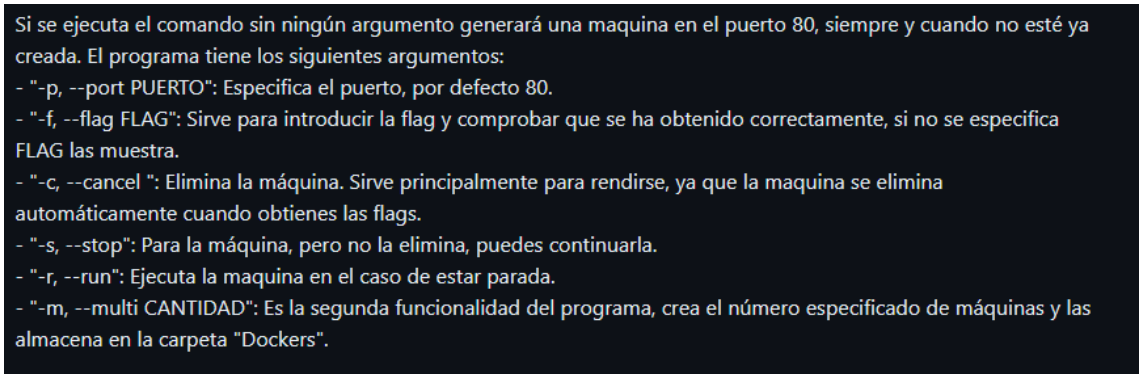

Imagen 6

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, se encarga de crear la máquina. Mas concretamente, la función “**crearNuevaMaquina**” 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 que cambian estos valores son “**establecerPuerto**”, “**establecerNombre**” y “**cambiarEstilo**” que cambian el puerto que especifiquemos (si lo especificamos) le dan un nombre (“altair” en caso de ejecutar el programa sin “-m”) y establecer un estilo para los CSS. También está “**anadirFlags**” que las introduce en la maquina y fuera de ella. Y “**crearUsuarios**” que modifica el archivo de la base de datos para añadirle una cantidad de usuarios aleatoria con una cantidad de atributos también aleatoria para que no sea fácil de predecir. Por último, tenemos la función “**elegirBdd**” y una función por cada posible motor de base de datos que tengamos programado, en un primer momento disponemos de “MYSQL”, “PGSQL” y “SQLite”, estas funciones elegirán un motor aleatorio y lo configurarán en la máquina.

También se le ha añadido las funciones "**cambiarEstiloInicio**" y "**crearEstructuraInicio**", estas funciones crean y le cambian el estilo a la pagina que te encuentras una vez estes logado. Estas funciones se ayudan de otras que podemos encontrar en esa parte del código.

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".

Imagen 7

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. 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.

5.1. 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, la autenticación del usuario carece de una implementación rigurosa, permitiendo la aceptación de credenciales débiles o fácilmente adivinables. 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.

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. La comunidad de seguridad cibernética reconoce la importancia de combatir activamente esta amenaza, enfocándose en medidas proactivas, educación continua y la implementación de buenas prácticas de desarrollo seguro.

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 un texto para que la respuesta sea siempre cierta. Por ejemplo, podemos introducir "**a' or 1=1 --**" (*Imagen $*) esto cierra el nombre de usuario después de la "a" y añade que el 1 sea igual a 1, siempre se cumplirá. Por último, añade "--" para que no de error con el resto de los caracteres. De contraseña se puede introducir cualquier carácter.

Habremos iniciado sesión con un usuario inventado, pero tendremos acceso al contenido de la web (*Imagen $*).

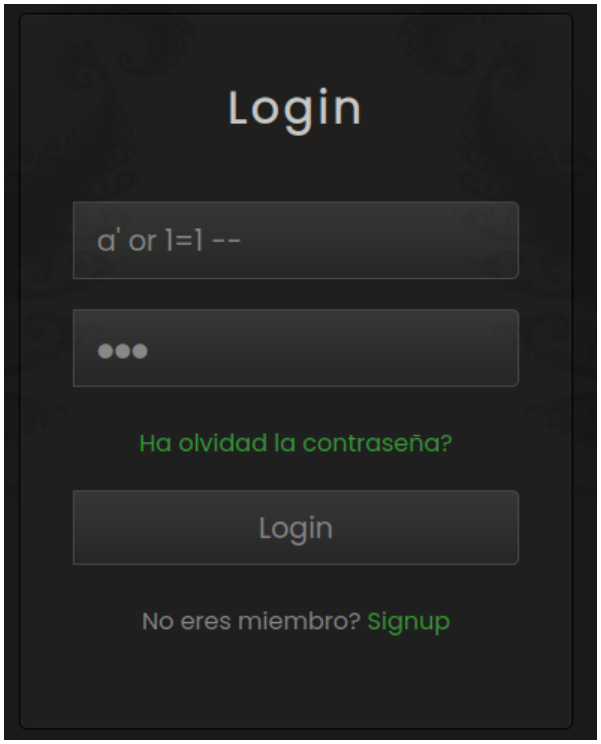

Imagen $

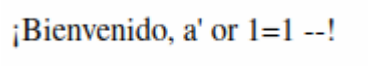

Imagen $

Esta vulnerabilidad tan típica se resuelve de la siguiente manera:

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.

5.2. 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.

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.

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

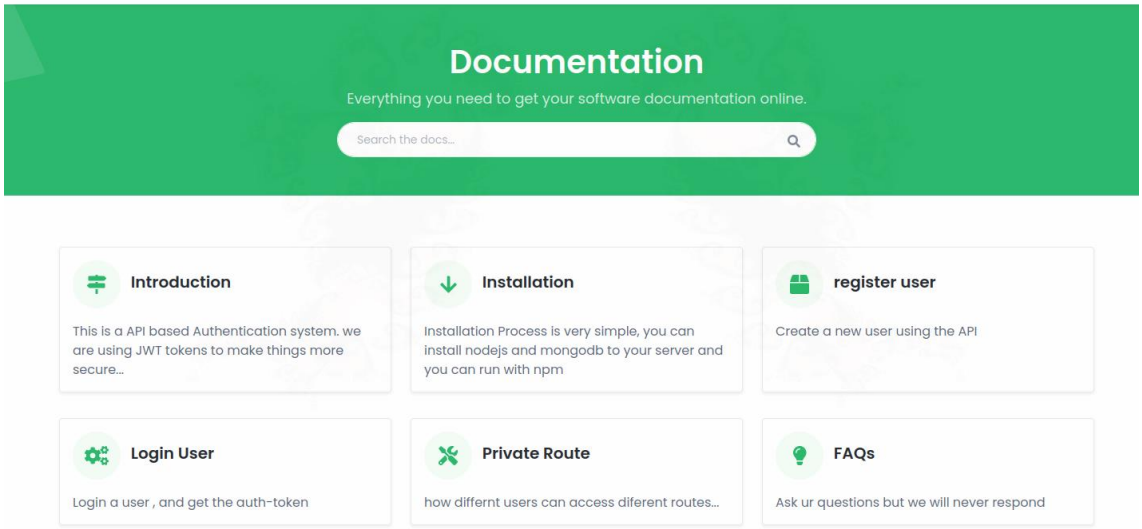

Imagen $

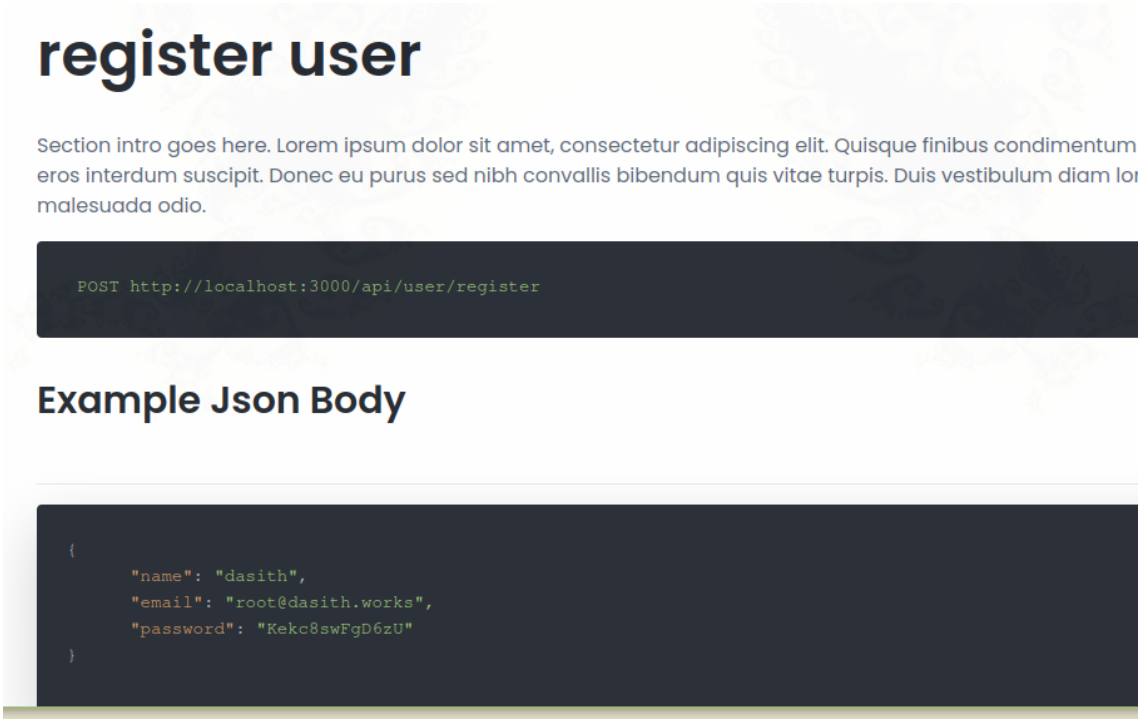

Imagen $

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 $*) con la única restricción que el nombre de usuario responda a "theadmin".

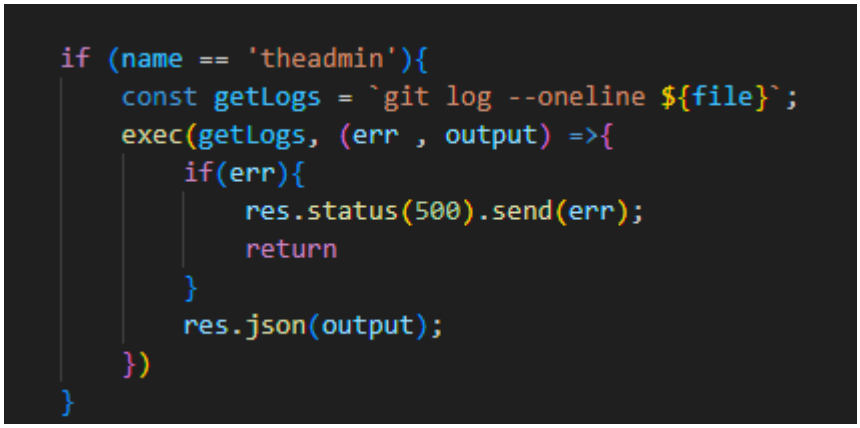

Imagen $

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 esta 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.

5.3. 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.

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 $*).

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 $

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.

Esta vulnerabilidad sirve 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 $*). 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.

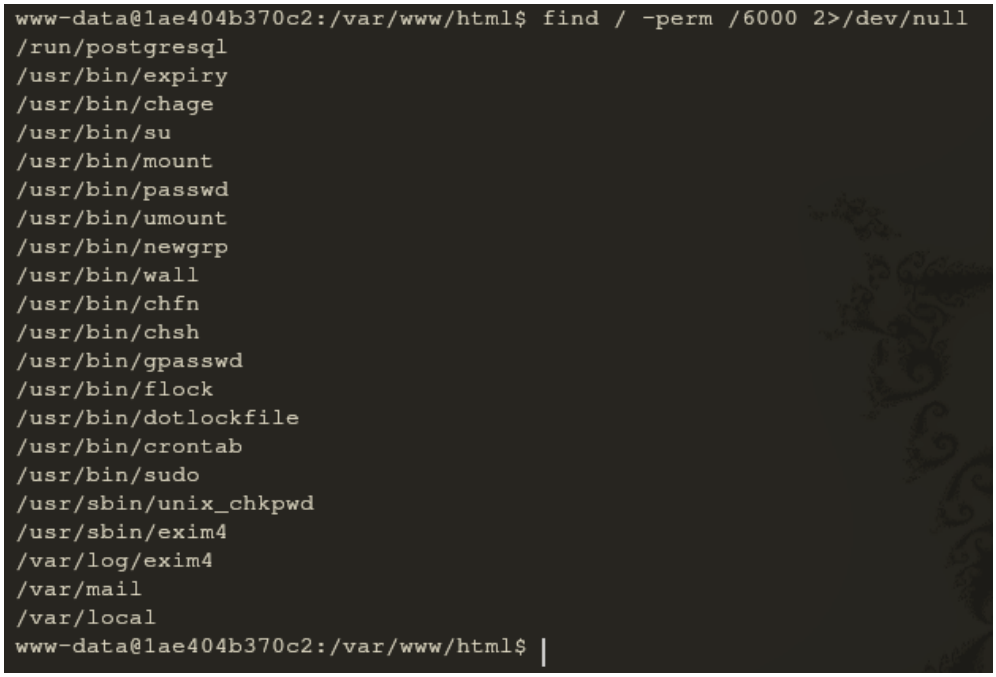

Imagen $

La mitigación de esta amenaza es muy parecida a la de “SUDO”, aun que 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.

6. CONCLUSIONES

El desarrollo de este proyecto ha proporcionado valiosas perspectivas sobre la ciberseguridad y el diseño de entornos educativos para la práctica de técnicas de hacking ético. Al finalizar este proyecto, se han extraído las siguientes conclusiones significativas:

6.1. Aprendizaje Activo en Ciberseguridad

La implementación de máquinas web simuladas con vulnerabilidades aleatorias ha demostrado ser un método efectivo para el aprendizaje activo en ciberseguridad. La combinación de aleatoriedad en las vulnerabilidades, la apariencia de la web y los datos de la base de datos ha proporcionado un entorno desafiante y realista que requiere adaptabilidad y resolución de problemas por parte de los participantes.

6.2. Importancia de la Aleatoriedad en la Formación

La introducción de aleatoriedad en diversos aspectos de las máquinas web ha destacado la importancia de la variabilidad en la formación en ciberseguridad. Esta característica refleja la diversidad de amenazas en entornos del mundo real y prepara a los participantes para abordar situaciones no predecibles.

6.3. Enfoque Estratégico en Vulnerabilidades Web

La distinción entre dos tipos de vulnerabilidades, centradas en la ejecución remota y la elevación de privilegios para administrador, ha permitido una comprensión más profunda de los desafíos específicos en el ámbito de la seguridad web. Aunque se ha reconocido la importancia de ambas categorías, el enfoque principal del proyecto en la ejecución remota ha proporcionado un marco claro para el desarrollo de habilidades específicas.

6.4. Reflexiones sobre la Seguridad Web:

Este proyecto ha arrojado luz sobre la importancia crítica de implementar prácticas robustas de seguridad web desde las etapas iniciales del desarrollo. La identificación y explotación de vulnerabilidades web no solo son habilidades técnicas, sino también un recordatorio constante de la necesidad de abordar la seguridad como una prioridad integral.

En conclusión, este proyecto no solo ha cumplido con sus objetivos educativos, sino que ha sentado las bases para futuras exploraciones en el vasto campo de la ciberseguridad. La combinación de aleatoriedad, enfoque estratégico y elección informada de tecnologías ha creado un entorno educativo dinámico y desafiante.

7. REFERENCIAS

GitHub del proyecto: https://github.com/ibantxu12/SSHKey_openldap

https://gtfobins.github.io/

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/