UNIMap: Sistema para la Detección y Gestión de Vulnerabilidades de Seguridad Utilizando Python, 2024

OBJETIVOS

Objetivo General

Desarrollar un sistema en Python que realice el escaneo, la identificación y la gestión de vulnerabilidades y problemas de conectividad en una red de dispositivos, con el fin de mejorar la seguridad de los sistemas y facilitar su mitigación.

Objetivos Específicos

1. **Desarrollar un módulo de escaneo de conectividad** que utilice python-nmap para realizar un escaneo de descubrimiento de hosts (utilizando el argumento -sn), permitiendo identificar dispositivos activos e inactivos dentro de un rango de IPs.

2. **Desarrollar un módulo de análisis de vulnerabilidades** que utilice python-nmap para realizar escaneos de puertos en dispositivos de la red. Este módulo debe identificar vulnerabilidades específicas asociadas a los servicios expuestos, clasificarlas según su nivel de riesgo (alto, medio, bajo), y proporcionar detalles sobre la mitigación de cada vulnerabilidad detectada.

3. **Diseñar un sistema de gestión de vulnerabilidades** que permita notificar a los administradores sobre las vulnerabilidades críticas encontradas. Estas notificaciones se enviarán por correo electrónico utilizando Gmail, con los reportes generados adjuntos (en formatos CSV, TXT y HTML), y proporcionando recomendaciones de mitigación basadas en buenas prácticas de ciberseguridad.

4. **Generar reportes detallados** sobre las vulnerabilidades y problemas de conectividad detectados, permitiendo la exportación de los resultados en formatos como CSV, TXT y HTML. Los reportes deben ser claros y proporcionar información relevante para que los administradores puedan tomar decisiones estratégicas sobre la seguridad de la red.

ANTECEDENTES

Pomachagua Sotomayor, J. L. (2020), en su tesis *Desarrollo de un Sistema de Auditoría de Equipos de Seguridad de Redes*, realizado en la Pontificia Universidad Católica del Perú, propuso un sistema que automatiza la auditoría de configuraciones de equipos de seguridad de redes como firewalls y directorios activos. El objetivo principal de esta investigación fue proporcionar una visibilidad clara y precisa del estado actual de los equipos de seguridad en una organización, con el fin de detectar y mitigar posibles vulnerabilidades.

Este sistema permitía verificar la correcta configuración de los dispositivos a través de interfaces API y protocolos como LDAP para el directorio activo, con lo cual se optimiza el análisis y la generación de reportes detallados sobre el estado de las políticas de seguridad implementadas en dichos equipos. La metodología empleada incluyó un enfoque práctico y automatizado que permitía realizar auditorías de manera continua y rápida, minimizando el margen de error humano y ofreciendo un control más eficiente sobre la infraestructura de red.

Entre las principales problemáticas abordadas se encontraba la creciente complejidad de gestionar los sistemas de seguridad de manera manual, dada la constante evolución de las amenazas informáticas y la necesidad de mantener los equipos actualizados y alineados con las mejores prácticas de seguridad, como la norma ISO 27001. Las pruebas del sistema se realizaron en entornos empresariales con directorios activos de más de 400 usuarios y firewalls con más de 1500 políticas de seguridad. Los resultados mostraron que el sistema fue efectivo al proporcionar a los administradores una herramienta que facilitaba el monitoreo y auditoría de las configuraciones seguridad con los requisitos indicados previamente., garantizando una infraestructura más segura.

En el estudio realizado por García Fernández, Á. (2020), titulado *Auditoría Automatizada Basada en un Sistema de Detección de Vulnerabilidades y en la Explotación Controlada de Amenazas Software*, llevado a cabo en la Universidad de Málaga, se desarrolló una aplicación con el propósito de fortalecer la seguridad del software en dispositivos mediante auditorías automatizadas y análisis de comportamiento del sistema. Este sistema proporcionaba una interfaz intuitiva que permitía a los usuarios monitorizar sus dispositivos de forma accesible, ayudando a identificar amenazas potenciales en tiempo real.

La investigación siguió un enfoque iterativo basado en la metodología SCRUM, lo que permitió un desarrollo ágil y eficiente para identificar errores conceptuales durante las fases de implementación. Los resultados demostraron que el sistema alcanzó sus objetivos al permitir a los usuarios sin conocimientos técnicos avanzados estar informados del estado de seguridad de sus dispositivos. Uno de los aspectos más innovadores fue la integración de YARA, una herramienta que permite escanear procesos en ejecución y detectar malware oculto mediante una base de reglas actualizable y flexible.

PROYECTOS SIMILARES

OWASP Nettacker (n.d.) es una herramienta de pruebas de penetración automatizadas escrita en Python y de código abierto, diseñada para ayudar a los profesionales de la seguridad a realizar auditorías de vulnerabilidad y reconocimiento de manera eficiente.

- Automatización de Pruebas de Seguridad: Realiza tareas automáticas de recopilación de información, escaneo de vulnerabilidades y fuerza bruta de credenciales.
- Soporte Multi-Protocolo: Escanea diversos protocolos como HTTP, FTP, SSH, SMTP, y más.
- Escáner de Puertos y Enumeración de Subdominios: Incorpora módulos integrados para detectar puertos abiertos y descubrir subdominios ocultos.
- Modular y Personalizable: Permite agregar nuevos módulos para tareas específicas y adaptarse a diversos escenarios de pruebas de seguridad.
- Generación de Informes en Múltiples Formatos: Ofrece informes en HTML, CSV, JSON y texto para facilitar el análisis y la integración de los resultados.

- Base de Datos Incorporada: Almacena los resultados de los escaneos, permitiendo el seguimiento y la recuperación de datos previos.
- Interfaz Web y API: Dispone de una interfaz web fácil de usar y una API para integración y automatización con otras herramientas.

Vulscan (n.d.) es un módulo que expande las capacidades de Nmap, convirtiéndo en un escáner de vulnerabilidades más robusto. Utilizando la opción `nmap -sV`, se detectan las versiones de los servicios, lo que permite identificar posibles fallas basadas en los productos detectados. Los datos se consultan a partir de una versión offline de la base de datos VulDB.

Plantilla de Informe Dinámico:

El siguiente conjunto de elementos puede ser utilizado en una plantilla para generar informes dinámicos:

- **{id}**: ID de la vulnerabilidad.
- **{title}**: Título de la vulnerabilidad.
- **{matches}**: Conteo de coincidencias.
- **{product}**: Cadena(s) de productos coincidentes.
- **{version}**: Cadena(s) de versión coincidentes.
- **{link}**: Enlace a la entrada correspondiente en la base de datos de vulnerabilidades.
- **\n**: Nueva línea.
- **\t**: Pestaña.

VULNR ⁇ PO - Generador y Repositorio de Informes de Vulnerabilidad (n.d.)

El proyecto utiliza un navegador para encriptar/desencriptar (AES) y almacenar los datos de forma local, garantizando así la confidencialidad total de los datos mediante encriptación de extremo a extremo. Por defecto, no se envía nada fuera del sistema. No utiliza un sistema backend, solo tecnología front-end con un cliente JS puro.

- Plantillas personalizadas para problemas:
  El uso de plantillas acelera significativamente el trabajo de los pentesters o auditores de seguridad. También es posible importar datos de CVE, CWE, MITRE ATT&CK o PCI DSS.

- Importación de problemas desde escáneres de seguridad:
  Es compatible con la importación desde los siguientes escáneres de seguridad: Nmap, Nessus, Burp, OpenVAS, Bugcrowd, Trivy, NPM, Semgrep, Composer. Tras la importación, se pueden gestionar y editar fácilmente las vulnerabilidades.

- Informes en múltiples formatos:
  Los informes pueden descargarse en formatos TXT, HTML, DOCX y PDF. También está disponible una versión HTML cifrada del informe. Si se necesita un PDF, basta con "imprimir como PDF" el informe en HTML.

- Archivos adjuntos:
  Puedes adjuntar fácilmente cualquier archivo que desees, como capturas de pantalla, videos o resultados del escáner en formato TXT. El sistema genera automáticamente un checksum SHA256 del archivo adjunto para verificar su integridad.

- Exportar problemas:
  Los problemas detectados se pueden exportar a sistemas de seguimiento de errores populares como Atlassian JIRA, o compartir de manera segura solo los problemas.

- Compartir informes:
  Puedes compartir tu informe utilizando encriptación AES por defecto, lo que garantiza la seguridad de los datos durante el proceso de compartición.

DISEÑO DE LA SOLUCIÓN

Problema Por Resolver

Las redes de dispositivos enfrentan desafíos para identificar y gestionar vulnerabilidades en sus infraestructuras de red. A menudo carecen de herramientas para realizar escaneos de seguridad, detectar puertos abiertos y recibir alertas oportunas sobre posibles riesgos de seguridad. Esto pone en peligro la seguridad de sus sistemas y aumenta la probabilidad de ser vulnerables a ataques cibernéticos.

Visión General de la Solución

El proyecto plantea desarrollar un Sistema de Gestión de Vulnerabilidades y Conectividad en Python, diseñado para ayudar a detectar, analizar y gestionar vulnerabilidades en una red de dispositivos, así como evaluar la conectividad de los dispositivos. Mediante el uso de técnicas de Programación Orientada a Objetos (POO), el sistema ofrece las siguientes funcionalidades:

1. Escaneo de redes para detectar dispositivos activos e inactivos.
2. Identificación y clasificación de vulnerabilidades asociadas a los servicios expuestos.
3. Generación de reportes detallados sobre el estado de la red y las vulnerabilidades encontradas.
4. Alertas por correo electrónico enviadas a los administradores de las vulnerabilidades críticas, con recomendaciones para mitigar los riesgos.

Componentes Principales

- **Interfaz de Usuario (UI):**

  - Proporciona una interacción intuitiva mediante una interfaz gráfica fácil de usar.
  - Permite configurar parámetros clave como: el rango de direcciones IP (en formato CIDR), el tipo de análisis a realizar (vulnerabilidades o conectividad), y el correo electrónico para recibir notificaciones.
  - También presenta los resultados de manera clara y estructurada para facilitar la interpretación.

- **Escáner de Red (NetworkScanner):**

  - Realiza escaneos de red utilizando python-nmap, detectando puertos abiertos y dispositivos conectados.

- Proporciona análisis de conectividad para determinar si los dispositivos están activos o inactivos.
- Identifica servicios en los puertos escaneados y evalúa vulnerabilidades.
- Implementa multiprocesamiento para acelerar los análisis en redes grandes.

- **Gestor de Vulnerabilidades (Vulnerability):**

  - Administra las vulnerabilidades identificadas durante los escaneos.
  - Utiliza un archivo JSON como base de datos para correlacionar servicios con niveles de riesgo conocidos.
  - Clasifica las vulnerabilidades en niveles de riesgo (bajo, medio, alto) y proporciona información detallada sobre las amenazas, incluyendo recomendaciones para mitigarlas.

- **Base de Datos de Vulnerabilidades (VulnerabilityDatabase):**

  - Carga y gestiona un archivo JSON que funciona como base de datos para almacenar información sobre servicios y posibles vulnerabilidades.
  - Es consultado directamente por el Gestor de Vulnerabilidades para identificar y correlacionar servicios detectados durante el escaneo con sus respectivos riesgos y recomendaciones.

- **Gestor de Conectividad (ConnectionIssues):**

  - Gestiona los resultados de conectividad detectados durante el escaneo de red.
  - Analiza el estado de los dispositivos en la red, identificando si están activos (en línea) o inactivos (fuera de línea).

- **Generador de Reportes (ReportGenerator):**

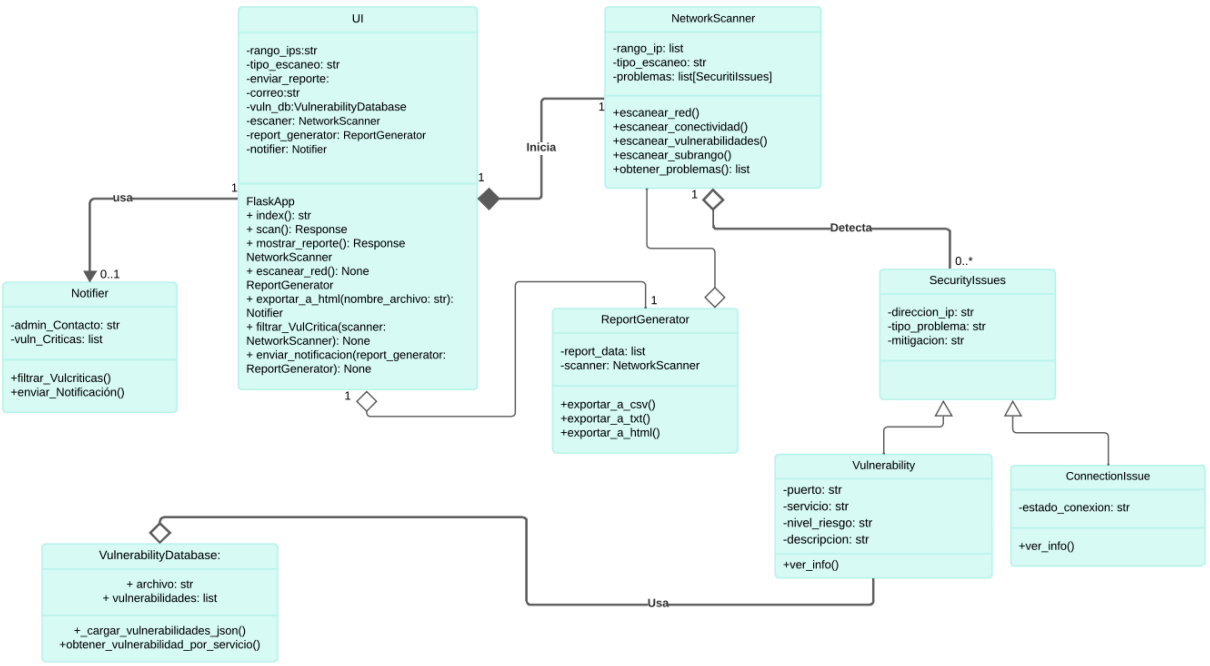
  - Compila los resultados del análisis en reportes organizados.
  - Permite exportar los reportes en formatos CSV, HTML y TXT, adaptándose a distintas necesidades.
  - Incluye información clave, como dispositivos analizados, vulnerabilidades detectadas, niveles de riesgo y recomendaciones.

- **Notificador (Notifier):**

  - Automatiza el envío de notificaciones en caso de identificar vulnerabilidades críticas.
  - Envía correos electrónicos a los administradores de red con información detallada sobre las amenazas detectadas y proporcionando acciones correctivas.

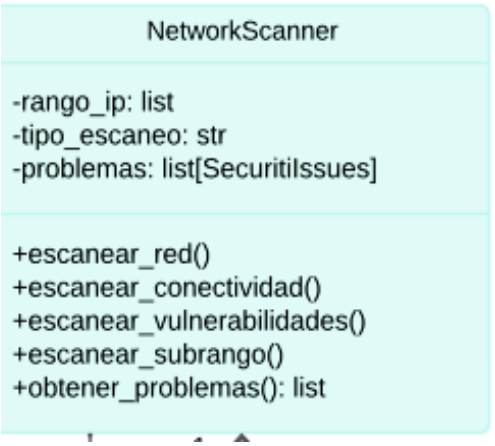
DIAGRAMA UML: Arquitectura del Sistema de Gestión de Vulnerabilidades

Este diagrama UML muestra cómo están organizados los componentes principales del sistema de **gestión de vulnerabilidades y conectividad**. Ilustra las relaciones entre las clases y cómo interactúan entre sí..



1. **Clase NetworkScanner:**

Realiza el escaneo de red y verifica la conectividad de los dispositivos.

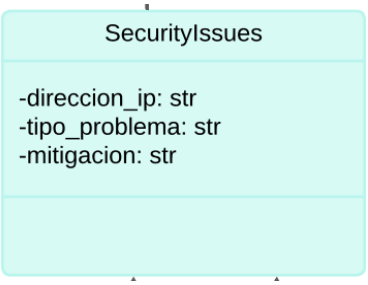


- **Atributos:**
  - `rango_ip`: Lista con el rango de IPs a escanear.
  - `tipo_escaneo`: Tipo de análisis (conectividad o vulnerabilidades).
  - `problemas`: Lista de problemas detectados (`SecurityIssues` y derivados)

- **Métodos:**
  - `escanear_red()`: Decide el tipo de escaneo a realizar.
  - `escanear_conectividad()`: Verifica conectividad en los dispositivos.
  - `escanear_vulnerabilidades()`: Busca servicios vulnerables dividiendo el rango en subredes.
  - `escanear_subrango()`: Analiza un subrango de IPs para servicios específicos.
  - `obtener_problemas()`: Devuelve la lista de problemas encontrados.

2. **Clase Security Issues:**

Clase base para representar problemas de seguridad.

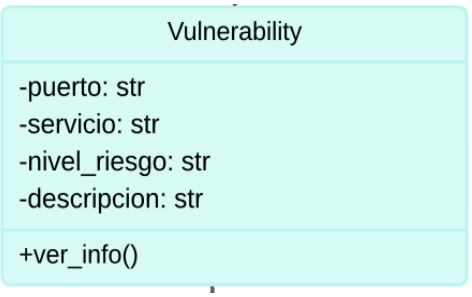


- **Atributos:**
  - `direccion_ip`: IP afectada.

- `tipo_problema`: Tipo de problema (conectividad o vulnerabilidad).
- `mitigacion`: Acción recomendada para resolver el problema.

3. **Clase Vulnerability:**

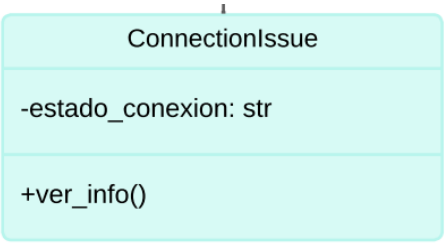
Gestiona las vulnerabilidades detectadas.



- **Atributos:**
  - `puerto`: Puerto afectado.
  - `servicio`: Servicio identificado.
  - `nivel_riesgo`: Nivel de riesgo (Alto, Medio, Bajo).
  - `descripcion`: Descripción de la vulnerabilidad.
- **Métodos:**
  - `ver_info()`: Devuelve detalles de la vulnerabilidad como diccionario.

4. **Clase ConectionIssues:**

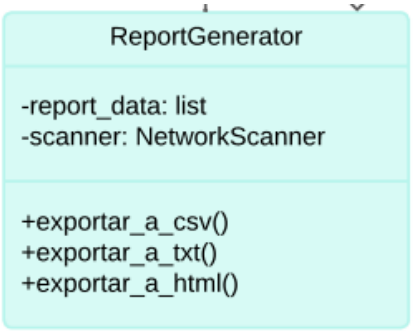
Analiza el estado de los dispositivos en la red, determinando si están activos o inactivos.



- **Atributos:**
  - `estado_conexion`: Estado de la conexión (activo o inactivo).
- **Métodos:**
  - `ver_info()`: Devuelve detalles del problema de conectividad como diccionario.

5. **Clase ReportGenerator:**

Se encarga de convertir los resultados del escaneo de red (proporcionados por el `NetworkScanner`) en reportes en formatos legibles: **CSV**, **TXT** y **HTML**.

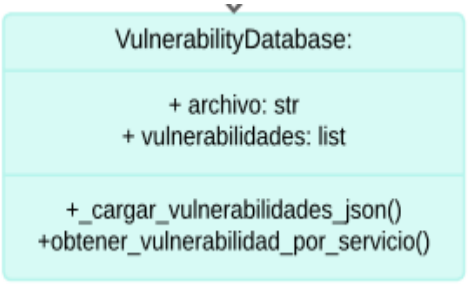


- **Atributos:**
  - `report_data`: Contiene los datos procesados del escáner de red, obtenidos a través del método `obtener_problemas` de `NetworkScanner`.
  - `scanner`: Referencia al objeto `NetworkScanner`, del cual se obtienen los problemas detectados.
- **Métodos:**
  - `exportar_a_csv()`: Genera un reporte en formato CSV con la información procesada.
  - `exportar_a_txt()`: Genera un reporte en formato TXT con los detalles de los problemas encontrados.

- `exportar_a_html()`: Genera un reporte en formato HTML con una tabla interactiva y botones para navegación.

6. **Clase VulnerabilityDatabase**

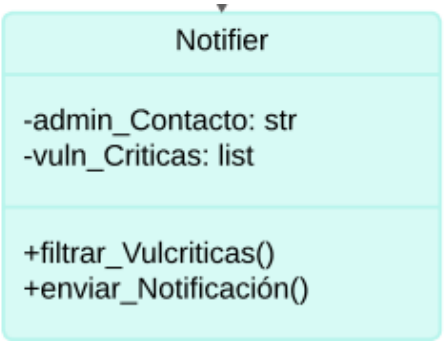
Almacena información sobre servicios y vulnerabilidades, correlacionando riesgos con los servicios detectados.



- **Atributos:**
  - `archivo:` Nombre del archivo JSON que contiene las vulnerabilidades.
  - `vulnerabilidades:` Lista de vulnerabilidades cargadas desde el archivo JSON.
- **Métodos:**
  - `_cargar_vulnerabilidades_json():` Método privado que lee el archivo JSON y carga la lista de vulnerabilidades.
  - `obtener_vulnerabilidad_por_servicio:` Devuelve la vulnerabilidad asociada a un servicio específico o un valor predeterminado si no se encuentra.

7. **Clase Notifier:**

Gestiona el envío de notificaciones por correo electrónico al administrador. Es útil para alertar rápidamente sobre problemas importantes detectados en el escaneo.



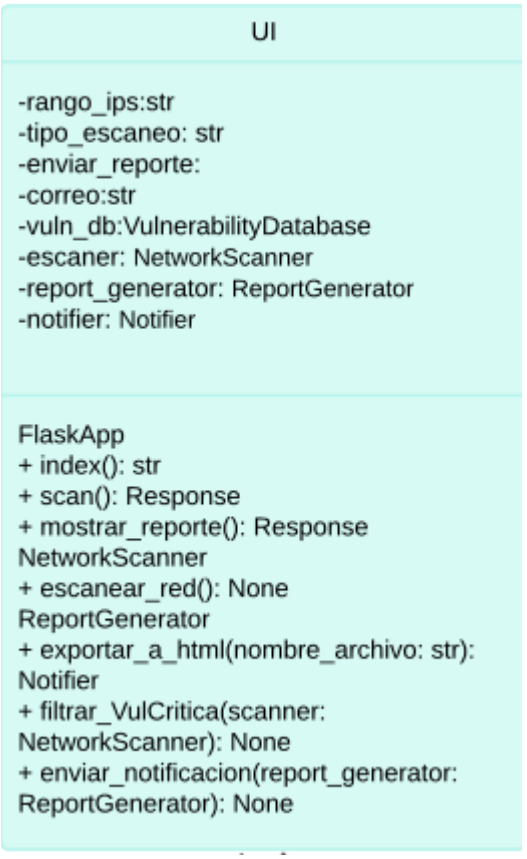
- **Atributos:**
  - `admin_Contacto:` Dirección de correo electrónico del administrador que recibirá las notificaciones.
  - `vuln_Criticas:` Lista de vulnerabilidades críticas filtradas durante el escaneo.
- **Métodos:**
  - `filtrar_Vulcriticas():` Filtra las vulnerabilidades críticas (niveles de riesgo "Alto" o "Medio") detectadas por el `NetworkScanner`.
  - `enviar_Notificacion():` Envía un correo electrónico al administrador usando una clave de aplicación con los reportes generados y un resumen de las vulnerabilidades críticas.

8. **Clase UI:**

Es la interfaz gráfica de la aplicación que conecta al usuario con la funcionalidad del escáner de red. Recoge los datos ingresados (rango de IPs, tipo de escaneo, y correo) y los utiliza para configurar los componentes principales: el escáner de red (`NetworkScanner`), el generador

de reportes (`ReportGenerator`), y el notificador (`Notifier`). Actúa como un controlador central que coordina estas funcionalidades.

- **Atributos:**
  - `rango_ips`: Rango de direcciones IP para el escaneo.
  - `tipo_escaneo`: Tipo de escaneo seleccionado por el usuario (conectividad o vulnerabilidades).
  - `enviar_reporte`: Indica si el reporte será enviado por correo (sí o no).
  - `correo`: Correo electrónico donde se enviará el reporte (si aplica).
  - `vuln_db`: Base de datos de vulnerabilidades para referenciar durante el escaneo.
  - `escanner`: Instancia del escáner de red que ejecuta el análisis.
  - `report_generator`: Generador de reportes para los resultados del escaneo.
  - `notifier`: Notificador para enviar alertas por correo.



FLUJO DE TRABAJO

1. Configuración Inicial:

   - Acción del Usuario: El administrador configura los parámetros iniciales del sistema, incluyendo el rango de direcciones IP a escanear, el tipo de escaneo a realizar (conectividad o vulnerabilidades), y la dirección de correo electrónico para recibir alertas (si es que se quiere enviar una notificación).
   - Entrada: Rango de IPs (en formato CIDR), tipo de escaneo, correo electrónico.

2. Inicio del Escaneo:

   - Acción del Sistema: El sistema comienza a realizar el escaneo de red. Dependiendo del tipo de escaneo seleccionado:
     - Si es un escaneo de conectividad, el sistema verifica si los dispositivos están activos o inactivos.
     - Si es un escaneo de vulnerabilidades, el sistema detecta los servicios expuestos (FTP, SSH, SMTP, etc.) y los puertos abiertos.
   - Proceso: El escaneo se realiza utilizando la biblioteca python-nmap para obtener información sobre los dispositivos en la red.

3. Análisis de Vulnerabilidades:

    - Acción del Sistema: Una vez completado el escaneo, el sistema analiza los servicios detectados y compara con su base de datos de vulnerabilidades (en formato JSON).
    - Proceso: El sistema identifica las vulnerabilidades asociadas a cada servicio y las clasifica en niveles de riesgo (alto, medio, bajo).

4. Generación de Reportes:

    - Acción del Sistema: El sistema genera reportes detallados en formato CSV, HTML y TXT, que incluyen:
        - Información de los dispositivos: IP, puerto, servicio.
        - Detalles de las vulnerabilidades detectadas: Descripción, nivel de riesgo y mitigación recomendada.
        - Acción del Usuario: El administrador puede exportar los reportes y revisarlos para tomar decisiones informadas.

5. Notificación de Vulnerabilidades Críticas:

    - Acción del Sistema: Si el sistema detecta vulnerabilidades de alto o medio riesgo, y el administrador ha habilitado la opción de enviar notificaciones durante la configuración inicial, se enviará automáticamente una notificación por correo electrónico a los administradores.
    - Proceso: El correo electrónico incluye una descripción detallada de las vulnerabilidades detectadas, el nivel de riesgo asociado y las recomendaciones para mitigar los riesgos.

6. Tareas Finales:

    - Acción del Usuario: El administrador revisa los reportes y las notificaciones. Con base en la información recibida, decide las acciones a tomar para mitigar las vulnerabilidades.
    - Entrada/Salida: Los datos de los reportes y las notificaciones se utilizan para mejorar la seguridad de la infraestructura de la red.

LIMITACIONES Y POSIBLES MEJORAS

Limitaciones Actuales:

1. **Cobertura de Servicios Limitada:**

    Actualmente, el sistema está diseñado para detectar vulnerabilidades en servicios comunes como FTP, SSH, Telnet, y SMTP. Sin embargo, no cubre todos los servicios posibles que podrían estar presentes en una red, especialmente servicios menos comunes o personalizados. Esto limita su capacidad de detección en entornos con configuraciones de red más complejas.

2. **Dependencia de Bases de Datos Externas:**

   El sistema depende de una base de datos de vulnerabilidades en formato JSON que debe ser mantenida manualmente. Cualquier desactualización de esta base de datos podría resultar en la omisión de vulnerabilidades importantes. Aunque se puede actualizar, esta dependencia externa puede ser un punto de fragilidad en el sistema.

3. **Escaneo Limitado de Conectividad:**

   El escaneo de conectividad solo verifica si los dispositivos están activos o inactivos, pero no evalúa otros aspectos de la red, como la latencia, la confiabilidad de la conexión o análisis de tráfico más profundos. Esto podría no ser suficiente en redes complejas o cuando se necesita un diagnóstico más exhaustivo.

4. **Falta de Integración con Otras Herramientas de Seguridad:**

   El sistema actualmente no está integrado con otras soluciones de seguridad de red o plataformas de monitoreo en tiempo real. La falta de integración podría limitar su efectividad en entornos donde se requieren herramientas de análisis de seguridad más avanzadas o sistemas de alerta en tiempo real.

5. **Dispositivos No Detectados por Firewalls o Protocolos Restrictivos:**

   Algunos dispositivos en la red pueden no ser detectados debido a que firewalls o protocolos restrictivos bloquean los intentos de escaneo. Esto puede suceder cuando los dispositivos están configurados para filtrar o rechazar solicitudes de red, lo que impide que el escáner obtenga información sobre ellos.

Posibles Mejoras:

1. **Ampliación de la Base de Datos de Vulnerabilidades:**

   Se podría expandir la base de datos de vulnerabilidades para incluir una gama más amplia de servicios y protocolos, lo que permitiría que el sistema detecte una mayor variedad de riesgos. Esto podría incluir la incorporación de vulnerabilidades relacionadas con nuevas tecnologías emergentes o servicios personalizados utilizados en la red.

2. **Automatización de Mitigaciones:**

   Actualmente, el sistema solo genera reportes y recomendaciones de mitigación. Una mejora sería la implementación de acciones automáticas de mitigación para corregir vulnerabilidades de manera inmediata, como cerrar puertos inseguros, deshabilitar servicios vulnerables, o aplicar configuraciones de seguridad recomendadas.

3. **Optimización del Rendimiento en Redes de Gran Escala:**

   Mejorar la eficiencia del sistema para escanear grandes rangos de IPs o redes más complejas. La integración de algoritmos más eficientes de escaneo y el uso de procesamiento distribuido podrían mejorar el rendimiento y reducir el tiempo de

escaneo en redes grandes.

4. **Interfaz de Usuario Más Avanzada:**
   Mejorar la interfaz gráfica para hacerla más intuitiva y accesible. Se podrían agregar más opciones de personalización para informes visuales, gráficos interactivos y un panel de control en tiempo real para monitorear el estado de la red.

BIBLIOGRAFÍA

- Pomachagua Sotomayor, J. L. (2020). *Desarrollo de un sistema de auditoría de equipos de seguridad de redes* [Tesis de licenciatura, Pontificia Universidad Católica del Perú]. Repositorio Institucional PUCP. https://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/19072
- Universidad de Málaga. (n.d.). *Desarrollo de un sistema de auditoría de seguridad*. Repositorio Institucional UMA. https://riuma.uma.es/xmlui/handle/10630/19202
- Gupta, J. (n.d.). *Python-Nmap-Scanner* [Repositorio de GitHub]. Recuperado de https://github.com/jaigupta9109/Python-Nmap-Scanner
- OWASP Foundation. (n.d.). *OWASP Nettacker* [Repositorio de GitHub]. Recuperado de https://github.com/OWASP/Nettacker
- SCIP. (n.d.). *vulscan* [Repositorio de GitHub]. Recuperado de https://github.com/scipag/vulscan

- Kac89. (n.d.). *vulnrepo* [Repositorio de GitHub]. Recuperado de https://github.com/kac89/vulnrepo/tree/master