UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de
Electrónica y Telecomunicaciones

Diseño e implementación de un *framework* para la evaluación periódica de la seguridad usando pruebas de penetración en la red interna de la Universidad de Cuenca.

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones.

Autores:

Brian Daniel Quinde Saltos
C.I: 030256325-9
Email: bdqs_2747@hotmail.com

Cinthya Lisseth Campoverde Andrade
C.I: 010651041-5
Email: cinthya.campoverde96@gmail.com

Director:

PhD. Darwin Fabián Astudillo Salinas
C.I: 010390703-6

Cuenca - Ecuador
26 de octubre de 2021

Resumen

La ciberseguridad en las Instituciones de Educación Superior (IES) se considera un factor trascendental debido a la información que manejan estas entidades, por lo que resulta de interés conocer constantemente el estado de la seguridad informática. En este trabajo de titulación se implementa un *framework* para el análisis periódico de la seguridad de la red de la Universidad de Cuenca mediante pruebas de penetración. El *framework* implementado consta de tres fases: obtención de información, explotación y reporte. Estas etapas están basadas en algunos de los procesos asociados a un *Red team*. Las herramientas que se usan en el *framework* son: The harvester, nuclei y NMAP. Estas herramientas se seleccionaron debido a que permiten la automatización del sistema y no precisan de interacción con el usuario. El análisis se debe realizar de forma periódica para lo cual se utiliza la herramienta Cron. El *framework* fue evaluado en un entorno controlado mediante el uso de una máquina virtual vulnerable. Luego se procedió a evaluar la seguridad de la Universidad de Cuenca. Las pruebas se ejecutaron desde dos escenarios. El primer análisis se realizó desde la red de Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia (CEDIA), y el segundo desde las instalaciones de la universidad. El análisis del reporte generado por la herramienta evidencia que existen vulnerabilidades en la red de la universidad que deben ser solventadas, para lo cual se emitió una recomendación por cada vulnerabilidad. Por lo tanto, la utilización del *framework* para el análisis periódico de vulnerabilidades ayuda a conocer el estado de la seguridad informática en la Universidad de Cuenca.

Palabras clave : *Framework*. Pruebas de penetración. *Hacking ético*. *Red team*. Detección de vulnerabilidades. Ciberseguridad.

Abstract

Cybersecurity in *Higher Education Institutions* (HEIs) is viewed as a trascendental factor because of the data took care of by these entities, so it is important knowing constantly the state of computer security. In this graduation project, a framework has been implemented for the periodic analysis of the network security of the University of Cuenca through penetration tests. The implemented framework comprises of three stages: acquiring data, exploitation and report. These stages depend on certain cycles related with a Red team. The tools used in the framework are: The harvester, nuclei and *Network Mapper* (NMAP). These tools were chosen since they permit the automatization of the system and do not need communication with the user. The analysis should be executed periodically, for which the Cron tool is utilized. The framework was assessed in a controlled environment utilizing a vulnerable virtual machine. Then, the security of University of Cuenca was assesed. The tests were run from two scenarios. The first analysis was produced using CEDIA's network, and the second from university premises. The analysis of the report created by the tool exhibits that there are vulnerability issues in the university network that should be solved, to which a suggestion was given for each vulnerability. Therefore, the utilization of the framework for regular analysis of vulnerabilities assists with knowing of the computer security status at University of Cuenca.

Keywords : Framework. Penetration tests. Ethical hacking. Red team. Vulnerability detection. Cybersecurity.

Índice general

Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Cinthya Lisseth Campoverde Andrade, autor/a del trabajo de titulación Diseño e implementación de un *framework* para la evaluación periódica de la seguridad usando pruebas de penetración en la red interna de la Universidad de Cuenca, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

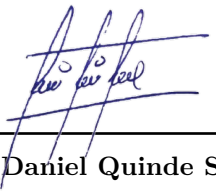
Cuenca, 26 de octubre de 2021

Cinthya Lisseth Campoverde Andrade
C.I: 010651041-5

Cláusula de Propiedad Intelectual

Brian Daniel Quinde Saltos, autor/a del trabajo de titulación Diseño e implementación de un *framework* para la evaluación periódica de la seguridad usando pruebas de penetración en la red interna de la Universidad de Cuenca, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Cuenca, 26 de octubre de 2021



Brian Daniel Quinde Saltos
C.I: 030256325-9

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Cinthya Lisseth Campoverde Andrade en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación Diseño e implementación de un *framework* para la evaluación periódica de la seguridad usando pruebas de penetración en la red interna de la Universidad de Cuenca, de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 26 de octubre de 2021

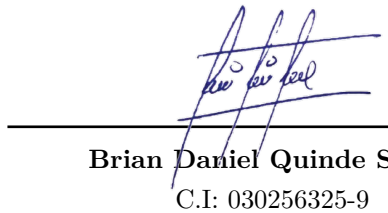
Cinthya Lisseth Campoverde Andrade
C.I: 010651041-5

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Brian Daniel Quinde Saltos en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación Diseño e implementación de un *framework* para la evaluación periódica de la seguridad usando pruebas de penetración en la red interna de la Universidad de Cuenca, de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 26 de octubre de 2021



Brian Daniel Quinde Saltos
C.I: 030256325-9

Dedicatoria

Esta tesis está dedicada a mi madre, por ser el pilar más importante y por enseñarme que con esfuerzo y dedicación todo se puede. A mi padre, por demostrarme siempre su cariño y apoyo incondicional. Ustedes han sido mi soporte a lo largo de mi carrera universitaria y de mi vida. A mis hermanos, Daniela y Sebastián, por estar siempre presentes y alegrarme en momentos difíciles. A mis tíos Vicente y Gladys, por ayudarme y darme consejos cuando lo necesitaba. A Paola, por escucharme y ser una amiga incondicional. Finalmente quiero dedicar esta tesis a mis abuelos, a pesar de nuestra distancia física, siento que están conmigo y sé que están orgullosos de mí.

Cinthya Campoverde A.

Dedicatoria

Esta tesis esta dedicada a:

A mi madre Ana quien con su cariño y paciencia me ha permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mi el ejemplo de esfuerzo y valentía.

A mi hermano Christian por su lealtad y apoyo incondicional durante todo este proceso y por estar conmigo en todo momento. A todos mis amigos por su lealtad y apoyo a lo largo de toda la carrera universitaria. A los docentes quienes inculcaron en mi el placer de aprender cada día un poco más.

Brian Quinde S.

Agradecimientos

Dice el escritor y científico estadounidense Isaac Asimov que “la ciencia puede divertirnos y fascinarnos, pero es la ingeniería la que cambia el mundo”, y así lo creemos nosotros ya que fue una de las cosas que aprendimos conforme avanzábamos en la carrera, y sirvió de motivación en el desarrollo de este trabajo de titulación.

Nos gustaría expresar nuestro más sincero agradecimiento al Ing. Darwin Fabián Astudillo Salinas, PhD, nuestro director, por su completo apoyo durante el desarrollo de esta tesis. También queremos agradecer a la Universidad de Cuenca, a la Facultad de Ingeniería, a nuestros docentes por brindarnos sus conocimientos y apoyarnos cuando lo hemos necesitado. Agradecemos también a nuestros amigos por brindarnos su apoyo durante el desarrollo de este trabajo.

Gracias a nuestros padres: Raúl y Rosa; y, Ana, por su amor, trabajo y sacrificio en todos estos años, debido a ustedes hemos logrado salir adelante y convertirnos en quienes somos. Finalmente, queremos agradecer a nuestros hermanos y hermanas por estar siempre presentes y por el apoyo moral que nos brindaron a lo largo de esta etapa de nuestras vidas.

LOS AUTORES

Abreviaciones y Acrónimos

AECI *Asociación Ecuatoriana de Ciberseguridad*. 21
API *Application Programming Interface*. 41, 68
ASP *Active Server Pages*. 70

BIESS Banco del Instituto Ecuatoriano de Seguridad Social. 21
BSI *Bundesamt für Sicherheit in der Informationstechnik*. 20

CEDIA Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia. 1, 7, 8, 21–23, 61, 69–71, 73
CEF Centro de estudios financieros. 26
CLARA Cooperación Latino Americana de Redes Avanzadas. 61
CLI *Command Line Interface*. 30
CRLF *Carriage Return and Line Feed*. 40
CSIRT *Computer Security Incident Response Team* . 21
CSRF *Cross-site Request Forgery*. 70
CVE *Common Vulnerabilities and Exposures*. 51, 57, 58, 65, 67
CVSS *Common Vulnerability Scoring System*. 56
CyC *Command & Control*. 37

DAST *Dynamic application security testing*. 41
DMZ *Demilitarized Zone*. 38
DNS *Domain Name Server*. 26, 31, 37
DOM *Document Object Model*. 41
DoS *Denial of service*. 65, 68, 69
DROWN *Decrypting RSA using Obsolete and Weakened Encryption*. 42

FMS *Fluhrer Mantin Shamir*. 33
FTP *File transfer protocol*. 30

GCF *Greenbone Community Feed*. 30
GPG *GNU Privacy Guard*. 53
GTR *Global Threat Intelligence*. 20
GVM *Greenbone Vulnerability Manager*. 7, 30, 31

HEIs *Higher Education Institutions*. 2
HSEEP *Homeland Security Exercise and Evaluation Program*. 27
HTTP *Hypertext Transfer Protocol*. 30, 31, 33, 69, 70

HTTPS *Hypertext Transfer Protocol Secure*. 33, 42

ICMP *Internet Control Message Protocol*. 7, 30, 46, 55
ID *Identification*. 36
IES Instituciones de Educación Superior. 1, 20, 21, 24, 42, 43, 72
IP *Internet Protocol*. 26, 29–31, 38, 42, 50, 51, 62, 66, 69
ISD *Intrusion Detection System*. 21
ISO *International Organization for Standardization*. 22, 43
ISSAF *Information System Security Assessment Framework*. 22, 39, 43
IT *Information Technology*. 21, 22
ITAC *Information Technology Association of Canada*. 39

JSON *JavaScript Object Notation*. 46, 51, 54, 73

LDAP *Lightweight Directory Access Protocol*. 35, 40

MAC *Media access control*. 62
MD5 *Message-Digest Algorithm 5*. 65
MITM *Man in the middle*. 33, 47
MTTC *Mean total time to contain*. 29
MTTP *Mean total time to obtain privileges*. 29

NMAP *Network Mapper*. 2, 7, 29, 30, 42, 46, 48, 52–57, 64, 67, 69, 72, 73
NTT *Nippon Telephone & Telegraph*. 20

OpenVAS *Open Vulnerability Assessment System*. 30, 35
OSINT *Open Source Intelligence*. 30
OSSTMM *Open Source Security Testing Methodology Manual*. 22, 39
OWASP *Open Web Application Security Project*. 31, 39, 43

PDF *Portable Document Format*. 7, 59, 60, 73
PHP *Hypertext Preprocessor*. 70
PoC *Proof of Concept*. 37
PTES *Penetration Testing Execution Standard*. 8, 39, 40, 43, 73
PTW *Pyshkin, Tews, Weinmann*. 33

RAE Real Academia Española. 24
RFID *Radio Frequency Identification*. 38

SCA *Scanner Control Application*. 41
SGSI Sistema de Gestión de la Seguridad de la Información. 22
SHA *Secure Hash Algorithm*. 65
SMB *Server Message Block*. 35, 64
SMTP *Simple Mail Transfer Protocol*. 60
SQL *Structured Query Language*. 40, 41, 68, 70
SSH *Secure SHell*. 35, 61, 64, 65, 67
SSL *Secure Sockets Layer*. 42

TCP *Transmission Control Protocol*. 7, 31–33, 46, 50, 56
TLS *Transport Layer Security*. 42
TOR *The Onion Router*. 42

UDP *User Datagram Protocol*. 7, 46, 50, 55, 56
UPS Universidad Politécnica Salesiana. 22
URL *Uniform Resource Locator*. 30, 66
USB *Universal Serial Bus*. 36

VNC *Virtual Network Computing*. 35
VPN *Virtual Private Network*. 21, 38

WEP *Wired Equivalent Privacy*. 33
WLAN *Wireless Local Area Network*. 43

XML *Extensible Markup Language*. 41, 46, 54, 73
XSS *Cross-site Scripting*. 41, 65

YAML *Yet Another Markup Language*. 57

ZAP *Zed Attack Proxy*. 31

Capítulo 1

Introducción

Este capítulo enmarca los desafíos presentados debido al incremento de ataques cibernéticos en Ecuador y los aspectos positivos que aporta el desarrollo de un *framework* para la evaluación de la seguridad en las IES. En primer lugar, se detalla la problemática vinculada al aumento de ciberataques (Sección 1.1). Luego, se ofrece una propuesta para resolver el problema y se presenta la orientación que dispone el trabajo de titulación (Sección 1.2). A continuación, se establece el alcance (Sección 1.3) y objetivos (Sección 1.4) de la tesis.

1.1. Identificación del problema

Internet y los datos se han convertido en la principal fuente de poder y riqueza de las naciones [5]. A causa de la revolución digital actualmente se genera información a partir de una diversidad de dispositivos como teléfonos inteligentes, computadores, sensores, etc [6]. Desde hace algunos años, los datos se han convertido en un activo de valor incalculable para las entidades empresariales y de gobierno [7].

Cuando surgieron las redes, estas no fueron creadas para ser seguras. La seguridad se ha ido incluyendo en los protocolos en función de los requerimientos. El incremento del número de ataques cibernéticos a nivel mundial ha suscitado un mayor interés por la seguridad cibernética. En el año 2016 la cadena hotelera Marriot International reportó la filtración de datos de 500 millones de clientes durante el lapso de dos años. Los datos filtrados correspondían a información de contacto, tarjetas de crédito, número de pasaporte, entre otros [8]. Además, se ha experimentado un crecimiento global de secuestros de información o *ransomware* de 350 % en 2018 respecto al 2017 según una investigación realizada por la firma internacional *Nippon Telephone & Telegraph* (NTT) *Security* [9]. Un ataque de este tipo de *malware* infectó alrededor de 300.000 computadoras en 150 países. Los atacantes exigieron el pago de un rescate a cambio de las claves para descifrar los archivos. En enero de 2019, la *Bundesamt für Sicherheit in der Informationstechnik* (BSI) informó de la filtración de información financiera, tarjetas de identificación y chats privados de políticos alemanes. Estos datos fueron finalmente publicados en línea por *crackers* [9]. La corporación NTT presentó su informe *Global Threat Intelligence* (GTR) del

año 2021 en donde destaca los tipos de *malware* más empleados en 2020, donde el 41 % corresponde a mineros, el 26 % a troyanos, el 10 % a gusanos y el 6 % a *ransomware*. Además, resalta que el 50 % de las empresas consideran como su principal enfoque de ciberseguridad la protección de los servicios en la nube [10].

De igual forma, los ataques cibernéticos se han incrementado en Ecuador [11]. Según el estudio realizado en el año 2016 por la compañía de seguridad informática ESET, el 45.6 % de las empresas ecuatorianas han sido víctimas de ataques, estos fueron usando *malware* [12]. En septiembre de 2019, la firma *Virtual Private Network* (VPN) *Mentor* reveló una filtración masiva de datos de más de 20 millones de personas en varias instituciones ecuatorianas. Entre las instituciones afectadas se encuentran el registro civil, así como cuentas y créditos del Banco del Instituto Ecuatoriano de Seguridad Social (BIESS). La violación de datos incluye información sensible de identificación personal a nivel individual [13]. A pesar de las graves amenazas a la seguridad en la red, según un análisis del estado actual de la ciberseguridad en Ecuador desarrollado por la Revista *Information Technology* (IT), en conjunto con la Consultora Deloitte y la *Asociación Ecuatoriana de Ciberseguridad* (AECI) en el año 2020, apenas un 20 % de las organizaciones participantes en el estudio, afirmaron realizar actividades de vigilancia para descubrir potenciales ataques y adoptar una estrategia de respuesta oportuna. Es importante enfatizar que más del 50 % de los participantes utilizan herramientas para reducir el riesgo de ciberseguridad tales como *firewalls*, *Intrusion Detection System* (ISD) y antivirus. Sin embargo, únicamente el 3 % cuenta con soluciones que permitan reducir riesgos relacionados con servicios de almacenamiento en la nube [14]. Según cifras de *Kaspersky Lab* por medio de su mapa de ciberamenazas en tiempo real, en el mes de mayo del año 2021, Ecuador ocupa el puesto 36 a nivel mundial y el cuarto puesto con respecto a América del Sur de los países que más ciberataques tienen a sus redes. El 74.59 % de estos fueron ocasionados por ataques de intrusión a la actualización de seguridad MS17010 disponible desde el sistema operativo Windows 7 en adelante [15]. Las empresas ecuatorianas han tenido que tolerar varios incidentes de seguridad que han significado enormes pérdidas, debido a que no están preparadas para afrontar los riesgos que surgen al adoptar las nuevas tecnologías [16].

Los datos presentados previamente evidencian que los ataques a la seguridad informática representan un riesgo a nivel global que debe ser priorizado con el fin de evitar consecuencias tales como la pérdida de información confidencial, daños financieros, interrupciones de servicios, filtración de datos, etc [17]. Las instituciones deben ofrecer un acceso seguro a sus servicios y prevenir infiltraciones de agentes externos o internos que puedan afectar a la información institucional [18]. Como cualquier otra institución, las IES tienen como prioridad proteger su información y brindar servicios seguros. Esta información consiste en datos de enseñanza, aprendizaje, investigación científica, gestión administrativa, datos culturales, etc [19]. Los ataques a las IES suelen dirigirse a la información crítica, por lo que los usuarios deben tener un plan de acción para salvaguardar los datos [20]. La violación de datos en una IES puede afectar gravemente sus intereses (reputación, operación y finanzas). Según el informe de actividades del año 2019 de la CEDIA, la operación del *Computer Security Incident Response Team* (CSIRT) de CEDIA logró disminuir, en un 31.84 % con relación al 2018, el número de vulnerabilidades, obteniéndose el valor más bajo desde 2015 [21]. Los datos presentados previamente sobre los riesgos que tienen las instituciones exponen la importancia de considerar temas de ciberseguridad para mitigar riesgos potenciales.

1.2. Justificación

Las pruebas de penetración son una parte importante del *Hacking* ético [22]. El proceso de prueba de penetración se puede definir como el estudio autorizado por las autoridades competentes, de las vulnerabilidades presentes en un cierto "objetivo" (servidores u ordenadores que manejan la información de una entidad). Este proceso de prueba concluye con la presentación de un informe sobre la seguridad del sistema, junto con pruebas de explotación que demuestren riesgos reales en los sistemas. En este punto se diferencia el *hacker* de sombrero blanco (miembro de un equipo *Red team*) de un *black* o *gray hacker*, ya que, en lugar de presentar un informe, estos actores maliciosos obtendrían beneficios de esas vulnerabilidades. Un concepto importante a tratar es el de *framework* de *hacking*. Un *framework* se puede definir como un conjunto de herramientas de software agrupadas bajo una interfaz común y permiten ejecutar tareas pertinentes a todas las fases del *hacking* ético. En este documento se utiliza el término *framework* para hacer referencia a un *framework* de *hacking* [23]. La norma *International Organization for Standardization* (ISO) 27001 recomienda que se deben realizar auditorías internas con intervalos planificados, para medir la efectividad de un Sistema de Gestión de la Seguridad de la Información (SGSI) de una institución. Una evaluación permite verificar que se cumplen los requisitos de la normativa, así como los requisitos específicos de seguridad de la organización [24]. En un artículo de la revista científica Dominio de las Ciencias por Diego Arcentales y Xiomara Caycedo se hace un enfoque efectivo acerca de la auditoría informática donde se determina que con esta, una empresa puede alcanzar estándares internacionales en el uso adecuado de ITs con miras a una certificación de calidad; así mismo, da cuenta del uso apropiado de los controles de riesgo de mayor impacto [25].

En la exploración previa del proceso de interés a nivel global se encontró una serie de documentos entre ellos una auditoría en la red de la Universidad Albaha usando una herramienta de penetración creada, donde se usa *Metasploit* y *Nessus* para encontrar la vulnerabilidad de un sistema [26]. Se encontró también un artículo de investigación que tiene el propósito concienciar acerca del *Hacking* ético, evaluando efectividad y eficiencia; se menciona también que estas pruebas son realizadas por pequeños grupos conocidos como *Red team*, los cuales explotan las vulnerabilidades de un sistema en un ambiente controlado, presentando riesgos reales que pudieran ser aprovechados por terceros [27]. En el trabajo de investigación [28] se presenta el progreso en la automatización de una serie de comandos para buscar e identificar las vulnerabilidades en los terminales, así como para explotarlas convirtiéndose en un proceso completo de *Red team*. A nivel nacional se encuentra la evaluación a la seguridad y la estructura implementada en la Universidad Politécnica Salesiana (UPS). Este trabajo se enfoca en realizar pruebas de penetración utilizando *Open Source Security Testing Methodology Manual* (OSSTMM), encaminado a la obtención de un diagnóstico de seguridad informática [29]. Otro trabajo de titulación en donde se trata sobre la identificación y análisis de vulnerabilidades en la sede de Quito de la UPS es el del autor Erik Parra; en este se realiza un análisis de vulnerabilidades utilizando pruebas de penetración siguiendo la metodología *Information System Security Assessment Framework* (ISSAF) [30]. La Universidad de Cuenca como muchas otras es miembro de CEDIA de donde se puede tener un precedente sobre la seguridad informática a forma de informe anual sobre IT presentado en el año 2019 donde se reporta la realización de 3 auditorías de seguridad a las redes externas y sitios web. Además, se menciona que se plantea la automatización de estas revisiones para realizarlas con mayor frecuencia [21].

Con el objetivo de abordar los desafíos planteados previamente, el presente trabajo de titulación

plantea el desarrollo de un *framework* para la evaluación periódica de la seguridad usando pruebas de penetración. Este trabajo tiene como objetivo realizar una evaluación periódica de las vulnerabilidades de seguridad en la red de la Universidad de Cuenca.

1.3. Alcance

El presente trabajo de titulación implementará una herramienta de pruebas de penetración utilizando Kali Linux. El *framework* desarrollado será empleado para evaluar las vulnerabilidades de seguridad existentes en la red interna de la Universidad de Cuenca. Las pruebas con el *framework* se realizarán desde la infraestructura de CEDIA. En la etapa inicial se realiza una evaluación de las herramientas que se van a integrar en el *framework*. A continuación; se implementan las fases del *Red team* descritas previamente. La fase de ataque se realiza en un clon de una máquina virtual perteneciente a la universidad para explotar el sistema sin que existan riesgos.

El desarrollo de esta herramienta pretende facilitar la realización de pruebas de penetración. También, se espera conocer el estado de seguridad informática mediante el uso de las siguientes métricas de seguridad: cobertura de aplicaciones web, frecuencia de las pruebas de penetración, tiempo de respuesta, criticidad de los hallazgos, tipos de vulnerabilidades y problemas solucionados [31]. Las métricas de seguridad descritas previamente serán utilizadas para la evaluación inicial.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar e implementar un *framework* de pruebas de penetración para evaluar de forma periódica la seguridad de la red de la Universidad de Cuenca.

1.4.2. Objetivos específicos

El presente trabajo tiene los siguientes objetivos específicos:

- Realizar un estado del arte sobre las herramientas utilizadas para el desarrollo de pruebas de penetración.
- Evaluar las diferentes herramientas de pruebas de penetración y seleccionar al menos tres de ellas para integrarlas en el *framework*.
- Implementar el *framework* de pruebas de penetración en la Universidad de Cuenca.
- Realizar pruebas con el *framework* desarrollado desde la infraestructura de CEDIA.
- Definir el formato de reporte que se presentará de manera periódica.
- Conocer el estado actual de la seguridad de la red de la universidad.

Capítulo 2

Marco teórico y trabajos relacionados

En este capítulo se da a conocer la base teórica necesaria para sustentar el presente trabajo de titulación. El capítulo inicia con una introducción hacia el *Hacking* ético (Sección 2.1), seguidamente se presenta el concepto de *Red team* (Sección 2.2) y se analizan sus fases y herramientas utilizadas. A continuación, se detallan las etapas de la cadena de eliminación (*Kill Chain*) (Sección 2.3) y se presenta la definición de pruebas de penetración (Sección 2.4). Luego, se detallan algunas de las vulnerabilidades empleadas para los ataques informáticos (Sección 2.5). El capítulo finaliza con la descripción de los trabajos relacionados a la implementación de *Hacking* ético en las IES (Sección 2.6).

2.1. *Hacking* ético

A medida que aumentan los ciberataques [32], también aumenta la demanda de profesionales de seguridad de la información con competencias en pruebas de penetración de red [33] y *Hacking* ético. El *Hacking* ético es una manera de hacer una evaluación de la situación actual de seguridad. Los resultados de este proceso es un informe detallado de los hallazgos, así como un testimonio de que un *hacker* ético en una cierta cantidad de tiempo y habilidades es o no capaz de atacar con éxito un sistema y/o obtener acceso a cierta información. La Real Academia Española (RAE) define al término *hacker* como *una persona experta en el manejo de computadoras, que se ocupa de la seguridad de los sistemas y de desarrollar técnicas de mejora* [34]. También, se define como "el arte de comprobar la existencia de vulnerabilidades de seguridad en una organización, para posteriormente, a través de un informe, revelar aquellos fallos de seguridad encontrados, mitigarlos a la brevedad posible y evitar fugas de información y ataques informáticos" [35]. Las personas que realizan el proceso de testeo se les designa como *hackers* éticos. Actualmente, se suele emplear el término *cracker* o intruso para los *hackers* que utilizan sus habilidades para el lado oscuro del *hacking*. En este proyecto de titulación se van a utilizar los términos *hacker* ético y cracker para describir los conceptos mencionados previamente.

Con el crecimiento de Internet, la seguridad informática se ha vuelto cada vez más importante para las organizaciones y los gobiernos. Estas organizaciones utilizan Internet en su amplia variedad de aplicaciones como el comercio electrónico, *marketing* y el acceso a bases de datos. Pero al mismo

tiempo, la seguridad de los datos y de las redes es un problema a tratar. Los datos privados disponibles en la red como los números de tarjetas de crédito, números de teléfono, direcciones personales, números de cuentas bancarias, etc., pueden ser *hackeados* por *crackers*. Inicialmente, las intrusiones en las computadoras fueron sin la intención de comprometer el sistema, pero actualmente se han convertido en un grave problema de seguridad.

Debido al aumento de ataques cibernéticos, las organizaciones decidieron realizar evaluaciones de seguridad realistas con el objetivo de reconocer cualquier intrusión en su red o sistema. Este proceso se realiza mediante sus propios profesionales capacitados que intentarían entrar en sus sistemas e identificarían si existe alguna amenaza de intrusión o vulnerabilidad. Estos profesionales, llamados *Red team* o *hackers* éticos, siguen los mismos pasos y herramientas que los *crackers*, pero se diferencian por sus intensiones. Los *hackers* éticos tienen intenciones de romper la seguridad informática para prevenir a la organización de ataques de intrusión. Ellos nunca revelan los hechos ni la información obtenida sobre la organización, pero si sus intenciones se desvían y se vuelven *crackers*; estos serían los que más daño causarían a la organización [1].

El *Hacking* puede ser clasificado en tres categorías, de acuerdo con los tonos del sombrero. La palabra sombrero tiene su origen en las películas del viejo oeste, donde el color del sombrero del héroe era blanco y del villano era negro. Por lo tanto, el término se ha desglosado en tres tipos presentados en la Figura 2.1.

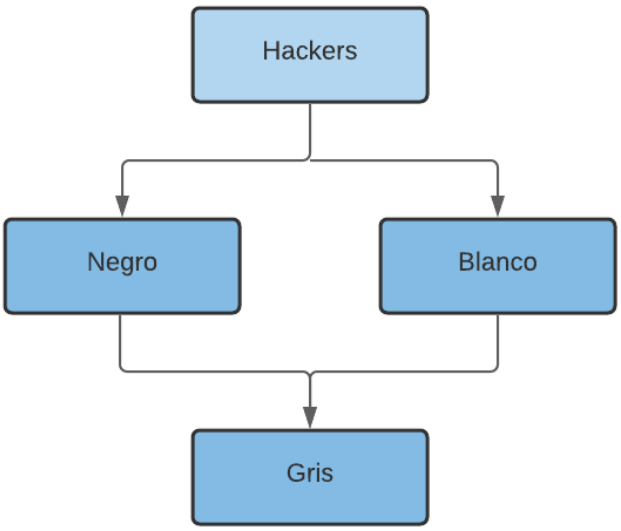


Figura 2.1: Tipos de *Hackers* [1]

1. ***Hackers* de sombrero blanco:** Personas autorizadas por la organización, con buenas intenciones e integridad moral. Su trabajo es proteger Internet, las organizaciones, las redes y los sistemas de los *crackers*. Algunas compañías les pagan para intentar *hackear* sus propios servidores y computadoras para probar su seguridad. Es decir, hacen *hacking* para el beneficio de la compañía. El *hacker* de sombrero blanco también es conocido como *hacker* ético [36].
2. ***Hackers* de sombrero negro:** La intención de estos *crackers* es dañar los sistemas informáticos y la red. Ellos rompen la seguridad e irrumpen en la red para corromper y destruir datos. Además,

pueden desfigurar las páginas web, roban los datos y violan la seguridad. Estos *crackers* dañan los programas y contraseñas para ingresar en la red o sistema no autorizado. Los *hackers* de sombrero negro ejecutan estas intrusiones por interés personal como el dinero. También son conocidos como *crackers* o *hackers* maliciosos.

3. ***Hackers* de sombrero gris:** Estos *hackers* recopilan información e ingresan en un sistema informático para romper la seguridad, con el propósito de notificar al administrador que hay fallas en la seguridad y el sistema puede ser *hackeado*. Entonces, pueden ofrecer a arreglar los problemas de seguridad existentes. Un sombrero gris puede violar la seguridad informática de las organizaciones y explotar sus sistemas pero suelen realizar cambios en los programas que pueden ser arreglados. Después de algún tiempo, informan al administrador sobre las brechas de seguridad existentes en la empresa. También, pueden *hackear* u obtener ingreso no autorizado en la red únicamente por diversión y sin la intención de dañar la red de las organizaciones [1].

Actualmente, el *Hacking* ético es considerado como una forma eficaz de fortalecer la seguridad de los sistemas informáticos en contra de ataques cibernéticos perpetrados por *crackers* [37]. Si una institución no conoce las fallas de seguridad en sus sistemas no es posible corregirlas lo cual la deja en un estado grave de vulnerabilidad. Un proceso de *Hacking* ético está organizado por fases que se fundamentan en metodologías de testeo, con el objetivo de ejecutar y alcanzar resultados de forma secuencial. Dentro de este ámbito, existen diversos procesos para el desarrollo de las fases. Rojas y colaboradores [38] se centran en un proceso compuesto de 6 fases: recopilación de información, escaneo, ataque, escalada de privilegios, instalación de puertas traseras y borrado de *logs*. En la primera fase se realiza la búsqueda de servidores *Domain Name Server* (DNS) y sus subdominios. En la fase de escaneo se realiza la detección de equipos vivos en la red, en la fase de ataque se adquiere información de bases de datos, credenciales, etc. La cuarta fase corresponde a la escalada de privilegios donde se pretende obtener privilegios para ejecutar código malicioso; en la fase de instalación de *backdoors* se instala un *backdoor*, el cual crea un canal oculto de comunicación entre dos *hosts* con el objetivo de realizar nuevos accesos en el futuro. Finalmente, en la fase de borrado de *logs* se limpian los registros del sistema generados cuando se realizó el ataque con el fin de eliminar posibles indicios del atentado.

Mendaño y Hurtado [39] implementan cinco fases reconocidas por el Centro de estudios financieros (CEF) las cuales son: reconocimiento, exploración, ganancia de acceso, mantener el acceso y borrar huellas. En la fase de reconocimiento se recoge información mediante el uso de varias herramientas. Se realizan dos tipos de reconocimiento: activo y pasivo. En el reconocimiento activo se obtiene información al realizar un análisis de la red para revelar *hosts*, direcciones *Internet Protocol* (IP), entre otros. En el reconocimiento pasivo se descubre información al acceder a la web e indagar acerca de la entidad. La fase de exploración a diferencia de la fase dos presentada en [38] emplea la información recolectada en la fase previa para realizar un análisis de vulnerabilidades. La tercera fase corresponde a la explotación de las vulnerabilidades previamente encontradas y la ganancia de privilegios para acceder al sistema. Entre los ataques utilizados frecuentemente se encuentran: desbordamiento de *buffer*, denegación de servicio, secuestro de sesión, ataques de fuerza bruta, entre otros. En la penúltima fase mediante el uso de *backdoors* se conserva el acceso con el objetivo de perpetrar otras intromisiones. Finalmente, en la fase de borrado de huellas es necesario eliminar la evidencia del procedimiento para conservar el acceso y eludir problemas legales.

Rojas y Ferney [37] se basan en un modelo de proceso de *Hacking* ético definido por el autor que consta de siete fases. En la primera fase (alcance de la prueba) se seleccionan los activos a valorar en el

proceso. Además, es necesario conseguir un permiso legal para el desarrollo de la intrusión para evitar problemas legales. En esta fase se detalla un proyecto considerando los sistemas a explotar, se evalúa un plan de contraataque en el caso de un ataque fallido, se establecen fechas y tiempos de ejecución de los ataques, así como las actividades a realizar en caso de encontrarse una vulnerabilidad crítica. En la fase de descubrimiento y enumeración se emplean procedimientos de reconocimiento activo y pasivo de forma similar a lo presentado por [39]. Estos procedimientos permiten recoger información valiosa para efectuar un ataque de forma exitosa. Entre las herramientas empleadas en la fase de descubrimiento están: WhoIs, Nslookup, Maltego, TraceRoute, entre otras. En la tercera fase (mapeo de vulnerabilidades) se realiza una búsqueda de vulnerabilidades. Existen dos formas de investigación de las vulnerabilidades: escenario interno que representa una búsqueda de las vulnerabilidades desde dentro de la organización y el escenario externo que investiga debilidades externamente. Para esta fase es común utilizar herramientas como Nessus, Retina, Microsoft Baseline Security Analyzer, etc. En la fase de ingeniería social se emplean personas con "habilidades sociales" para efectuar actividades malintencionadas. Algunas herramientas utilizadas para este proceso son: vigilancia, *pishing*, robo de identidad, etc. En la quinta fase (explotación) se ingresa al sistema al ejecutar un código *exploit* que aprovecha las vulnerabilidades encontradas en el sistema y las explota. Otro proceso necesario en esta fase es la obtención de privilegios con el objetivo de controlar de forma remota una máquina para tener acceso a todos los datos de la misma. En la penúltima fase (mantenimiento de acceso) de igual forma que lo descrito por [39] se ejecuta un *backdoor* para ingresar al sistema cuando sea necesario. Algunos programas empleados en esta fase son troyanos y *backdoors*. En la fase final denominada documentación y reportes se analiza la información recopilada y se presenta un informe detallado. En el caso de ser requerido es posible ofrecer una directiva con la corrección de las fallas de seguridad presentadas. Estas correcciones pueden realizarse de manera automática por un *software* de *hacking* o de forma manual.

Al realizar las pruebas de penetración como parte del *Hacking* ético se deben priorizar las consideraciones legales como la confidencialidad y que la información sea verídica para asegurar la fiabilidad del análisis [39]. El proceso de pruebas se analiza en la Sección 2.4. Para desarrollar un proceso de *Hacking* ético, el *Red team* debe ejecutar varias pruebas de penetración [40] para identificar cualquier amenaza que pueda dañar al sistema. A diferencia del *Red team*, el *Blue team* necesita garantizar que los objetivos de ataque estén seguros y en el caso que el *Red team* encuentre una vulnerabilidad y la explote, ellos deben remediarla inmediatamente y documentarla como parte de una lección aprendida. Este proyecto de titulación se enfoca en las actividades realizadas por un *Red team*.

2.2. *Red team*

Muchos términos usados en la seguridad de la información se originaron en el ejército. Los conceptos de *Red team*, o equipo rojo, y *Blue team*, o equipo azul fueron introducidos durante la primera guerra mundial. En seguridad informática y protección de datos se consideran los equipos *Red team* y *Blue team* [41]. Estos equipos ejecutan un desarrollo conjunto con el objetivo de hallar vulnerabilidades, evitar ataques y simular un escenario de ataque [42]. La idea de generar estos equipos es mostrar la efectividad de un ataque a través de simulaciones. La efectividad de las simulaciones se basa en tácticas reales que pueden ser utilizadas por un *cracker*. El *Homeland Security Exercise and Evaluation Program* (HSEEP) utiliza el *Red team* en los ejercicios de prevención para rastrear cómo los *crackers*

se mueven y desarrollar contra medidas basados en los resultados obtenidos de estos ejercicios.

En el campo de la ciberseguridad, la adopción del enfoque del *Red team* ayuda a las organizaciones a mantener sus activos más seguros. Este equipo debe estar compuesto por personas entrenadas con diferentes conjuntos de habilidades, y además, deben estar conscientes del panorama de amenazas existente para las organizaciones. El *Red team* debe saber las tendencias y entender cómo se desarrollan actualmente los ataques. En algunas circunstancias y dependiendo de los requerimientos de la organización, los miembros del equipo deben poseer habilidades de codificación para crear sus propios ataques y personalizarlos para explotar de mejor forma las vulnerabilidades relevantes que podrían afectar a la organización. El flujo de trabajo de un *Red team* sigue el enfoque presentado en la Figura 2.2.

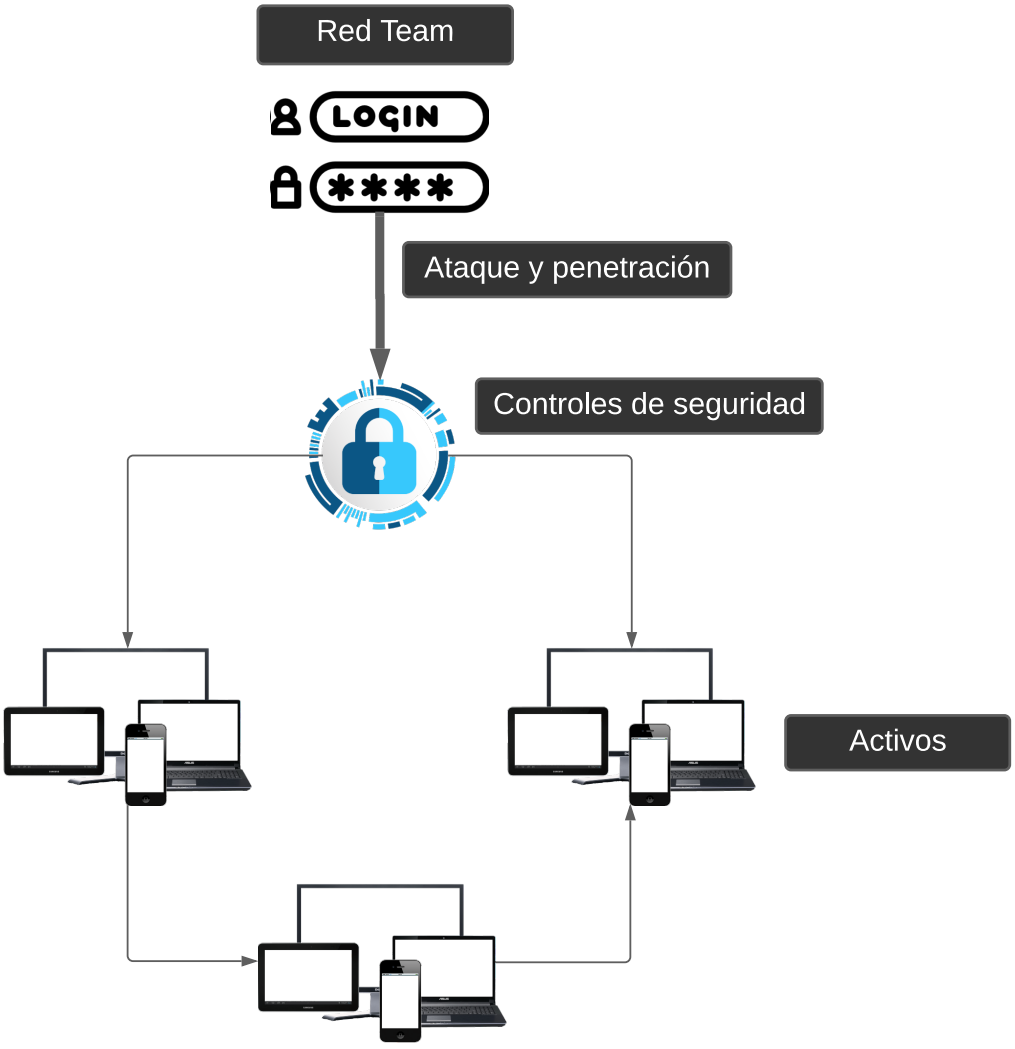


Figura 2.2: Flujo de trabajo de un *Red team* (Creación propia)

El *Red team* desarrolla un ataque e ingresa al entorno tratando de superar los controles de seguridad, también conocido como prueba de penetración. La intención de la misión es (1) exponer las vulnerabilidades de un objetivo, (2) degradar, desconectar, o denegar la habilidad de un usuario de acceder a cierto ambiente virtual, (3) probar las habilidades y estrategias del departamento de defensa, y (4) brindar soporte de operación en encuestas de seguridad informática [43]. Para medir el estado de

la seguridad informática de una entidad el *Red team* utiliza las siguientes métricas:

1. ***Mean total time to contain* (MTTC):** Esta métrica contabiliza desde el minuto que el *Red team* inicia el ataque hasta que son capaces de conseguir acceso al objetivo.
2. ***Mean total time to obtain privileges* (MTTP):** Esta métrica inicia al mismo tiempo que la métrica MTTC, pero sigue contabilizando hasta comprometer totalmente el objetivo, es decir; hasta que el *Red team* tenga permisos administrativos del objetivo.

El trabajo realizado por el *Red team* no termina cuando se logra comprometer el sistema; el *hacker* ético también debe crear un informe final que presenta los detalles acerca de la forma que se produjo el ataque, la cronología del mismo, así como los detalles de las vulnerabilidades explotadas para obtener el acceso y elevar privilegios [41]. Las fases de ataque y penetración generalmente siguen los lineamientos presentados en *"Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains"* [44]. Siguiendo estos lineamientos se hace uso de las fases descritas en las secciones a continuación.

2.2.1. Recopilación de información

En esta fase un atacante realiza la búsqueda de objetivos para conseguir la mayor cantidad de información posible desde fuera de la red. Esta información puede ser entre otras cosas fuentes de potencia, dispositivos antiguos utilizados y comportamiento en redes sociales de los empleados. Todo lo que se necesita de esta búsqueda es un punto débil que puede servir de entrada a la red de la organización. En esta fase existen dos técnicas comúnmente utilizadas las cuales son *phishing* e ingeniería social [41]. No obstante, se usan herramientas especializadas en el reconocimiento de puertos y vulnerabilidades existentes en la red objetivo.

2.2.1.1. *Nessus*

Esta herramienta puede escanear una red en búsqueda de vulnerabilidades y a su vez mostrar los dispositivos conectados que tengan configuraciones erróneas o actualizaciones de seguridad no instaladas. La herramienta también presenta los dispositivos que estén utilizando sus contraseñas predefinidas, débiles o que no tengan contraseñas del todo [41]. Nessus puede hacer uso de herramientas de terceros para muchos propósitos. Hoy en día existen varias versiones de Nessus, cada una con un propósito específico.

Nessus profesional automatiza las evaluaciones mediante plantillas creadas previamente. Esta herramienta agrupa las vulnerabilidades bajo muchos factores permitiendo priorizar tareas. Por último, tiene la capacidad de visualizar los resultados en tiempo real [45].

Nessus essential realiza un escaneo de hasta 16 direcciones IP. Sin embargo, no permite realizar verificaciones de cumplimiento, auditorías de contenido y visualización de resultados en tiempo real.

2.2.1.2. NMAP

NMAP es una herramienta de código abierto basada en Unix. Esta herramienta es mayormente utilizada para escanear redes en búsquedas de *hosts* activos, y servicios o sistemas específicos en operación. Parte de esta utilidad es la de crear paquetes IP desde un análisis superficial, enviarlos mediante métodos únicos y hacer escaneos [46].

NMAP tiene la capacidad de realizar escaneos básicos de la red con un *ping* de tipo ICMP para determinar si los *hosts* están activos o no. También, tiene la capacidad de conducir búsquedas avanzadas con múltiples opciones y escaneo en un amplio espectro de direcciones IP mientras se registran tipos de archivos o sistemas específicos [46]. NMAP es utilizando mediante líneas de comando; gracias a lo cual los usuarios pueden crear *scripts* personalizados [46].

2.2.1.3. The Harvester

The Harvester es una herramienta diseñada para ser utilizada en las primeras fases de una prueba de penetración o de una auditoría realizada por un *Red team*. Esta herramienta de *Open Source Intelligence* (OSINT) es utilizada para determinar el panorama de amenazas externas de la empresa en Internet. La herramienta obtiene correos electrónicos, nombres, subdominios, IPs y *Uniform Resource Locators* (URLs) mediante el uso de múltiples fuentes de información como Anubis-DB, Baidu, Bing, Censys, Google, Shodan y Trello [47].

2.2.1.4. Shodan

Shodan permite al usuario realizar consultas filtradas para encontrar los dispositivos conectados a Internet. Estos pueden variar desde dispositivos de domótica hasta sistemas de control de centrales térmicas [48]. también con el uso de esta herramienta, se puede saber el número de servidores conectados en un país y ciudad en específico. La información que Shodan proporciona sobre los dispositivos utiliza lo que se conocen como *banners*. Un *banner* es información textual (metadatos) sobre servicios que se ejecutan en los dispositivos. Existen diferentes tipos de *banners* que se encuentran relacionados con el tipo de servicio tales como el *Hypertext Transfer Protocol* (HTTP), Telnet, *File transfer protocol* (FTP). Shodan recoge otra información sobre los dispositivos como el *hostname*, la posición geográfica, el sistema operativo, la organización a la que pertenece entre mucha más información [48].

Shodan puede ser utilizado desde su sitio web, así como desde la consola de comandos de Linux, mediante la instalación de Shodan *Command Line Interface* (CLI). La diferencia principal radica en el costo computacional por cada búsqueda.

2.2.1.5. OpenVAS

Open Vulnerability Assessment System (OpenVAS) es un módulo (contenido en el servicio GVM, Figura 2.3) que contiene varias herramientas para ofrecer un escaneo profundo de vulnerabilidades y la aportación de soluciones. OpenVAS es una herramienta de código abierto que utiliza una base de datos comunitaria llamada *Greenbone Community Feed* (GCF). Esta base de datos es actualizada diariamente y tiene más de 50000 pruebas para realizar en las redes o equipos a analizar. En caso de necesitar un servicio más avanzado, se puede contratar el producto *Greenbone* más adecuado para la empresa o entidad, para tener acceso al servicio GVM [48].

2.2.1.6. Nikto

Nikto es una página web de escaneo de vulnerabilidades basada en Linux usada para identificar cualquier brecha de vulnerabilidad en los sitios web de organizaciones. También escanea en los servidores cerca de 6800 vulnerabilidades comunes explotables. La herramienta también escanea por versiones no parchadas de entre cerca de 250 plataformas. Nikto se puede utilizar desde una consola de comandos.

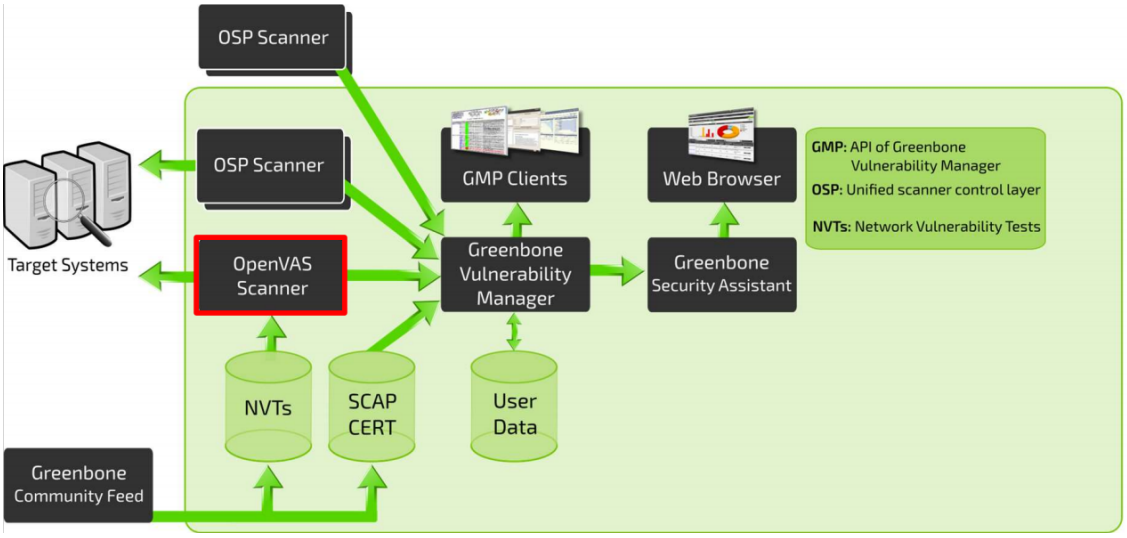


Figura 2.3: Arquitectura global de GVM [2]

El usuario da la dirección IP de la red a escanear y la herramienta presenta toda la información acerca del servidor web donde se aloja [41].

2.2.1.7. Nuclei

Esta herramienta se utiliza para enviar solicitudes a través de objetivos basados en una plantilla que conduce a cero falsos positivos y proporciona un escaneo rápido en un gran número de hosts. Nuclei ofrece escaneo para una variedad de protocolos incluyendo TCP, DNS, HTTP, archivo, etc. Esta herramienta permite ejecutar todo tipo de controles de seguridad. Nuclei tiene un repositorio dedicado que alberga varios tipos de plantillas de vulnerabilidad aportadas por más de 200 investigadores e ingenieros de seguridad [49].

2.2.1.8. OWASP

Open Web Application Security Project (OWASP) es un proyecto de código abierto diseñado para mejorar la seguridad de diferentes aplicaciones y servicios web. Con esta herramienta, se hacen públicos los resultados de diferentes análisis de terceros. Uno de los componentes más potentes de OWASP es el *Zed Attack Proxy* (ZAP). Esta es una plataforma diseñada para monitorear la seguridad de las aplicaciones web. Tiene la posibilidad de comprobar peticiones y respuestas entre cliente/servidor, localizar recursos en un servidor, y lanzar varios ataques a la vez [50]. Finalmente, una de las ventajas más grandes de esta herramienta es que permite trabajar de forma local [51].

2.2.1.9. Wireshark

Wireshark es una herramienta que captura paquetes en la red objetivo y los presenta en forma legible. La herramienta permite analizar el tráfico en la red al nivel de inspeccionar cada paquete individual. Wireshark funciona en dos modos. En el primer modo puede capturar paquetes por un periodo largo de tiempo. En el segundo modo la captura de datos puede ser detenida para proceder con un análisis más profundo. Una funcionalidad importante de la herramienta permite al usuario

visualizar las comunicaciones entre computadores [41].

2.2.2. Explotación

Una vez encontradas las vulnerabilidades del sistema objetivo, se tiene que explotar cada posible vulnerabilidad. Esto se hace mediante la investigación de las tendencias actuales en cuanto a las herramientas, técnicas y objetivos utilizados por *crackers*; de esta manera se puede obtener acceso al sistema. Las credenciales de usuario se utilizan ahora para comprometer el sistema o red objetivo [41]. Para aprovechar las vulnerabilidades encontradas se usan *exploits*. El término *exploit* hace referencia a la ejecución de ciertas acciones que hacen posible abrir una puerta del sistema, para la posterior inyección de acciones que permitan modificar o realizar una tarea sobre un objetivo vulnerable. Sin embargo, la acción de aprovechar una vulnerabilidad y la ejecución de comandos está diferenciada mediante dos términos: *exploit* y *payload*. Se utiliza un *exploit* para conseguir acceso a un sistema e inyectar acciones que permitan modificar u obtener información sensible. Mientras que, un *payload* contiene las instrucciones que lanzan una conexión TCP con el objetivo.

Las conexiones de tipo TCP utilizadas para la apertura de *shells* remotas se dividen en dos tipos: *bind* y *reverse*. Las conexiones tipo *bind* lanzan una petición directa al sistema vulnerable para abrir la conexión. Mientras, en la conexión de tipo *reverse* se habilita un puerto que escucha peticiones entrantes en la máquina *cracker* y se utiliza un *exploit* con *payload* para que sea el sistema vulnerable quien establezca la conexión hacia el sistema atacante. Para esto, el atacante usa herramientas que le permitan aprovechar las vulnerabilidades encontradas[41].

2.2.2.1. Metasploit

Este *framework* está compuesto por un número de herramientas que son utilizadas para escanear y explotar vulnerabilidades existentes en la red. Debido a las amplias capacidades de esta herramienta, muchos *hackers* de sombrero blanco lo usan para impartir información a estudiantes. También califica como una herramienta para realizar pruebas de penetración, siendo utilizada por una gran cantidad de empresas. Metasploit tiene un gran número de *exploits* que pueden ser utilizados en contra de navegadores, sistemas operativos Android, Microsoft, Linux y Solaris, además de muchos *exploits* aplicables a otras plataformas.

La ventaja de Metasploit radica en que posee mecanismos que detectan y evaden programas de seguridad que pueden estar presentes dentro de la red. La herramienta tiene una gran cantidad de comandos que pueden ser utilizados para rastrear información de la red. También, posee herramientas complementarias que pueden ser utilizadas para explotar las vulnerabilidades obtenidas en la fase de reconocimiento [41].

Metasploit ofrece un nivel de abstracción respecto al código de bajo nivel. El usuario final debe tener un conocimiento general para comprender como y cual es la vulnerabilidad objetivo (inyección de código, condición de carrera, escala de privilegios, entre otros), forma en la que se va a realizar la conexión con el sistema o *software*, y cuales son los módulos Metasploit destinados a explotar cierta vulnerabilidad. Metasploit es una herramienta desarrollada con el lenguaje de programación Ruby lo que hace posible manejar los módulos existentes, duplicar un módulo ya existente, modificar su nombre, insertarlo en el directorio de Metasploit y hacer uso de la herencia e importación de objetos para invocar funcionalidades existentes en la herramienta, sin la necesidad de crear *scripts* nuevos.

Metasploit ofrece la inyección de un *payload* sobre el sistema vulnerable mediante lo que se conoce como Meterpreter. Meterpreter es una *shell* avanzada con un mayor número de funcionalidades para ejecutar acciones complejas sobre el sistema objetivo, provee sigilo para no dejar una traza del ataque realizado y mantiene las comunicaciones cifradas entre el sistema vulnerable y el *cracker* [52].

2.2.2.2. Armitage

Armitage es una herramienta que funciona con una estructura de cliente/servidor y permite la colaboración como *Red team*, pues posibilita el uso de *scripts* para Metasploit, visualizar objetivos y recomienda *exploits* [50]. Armitage organiza las capacidades de Metasploit. Permite descubrir, acceder, post explotar y manejar los sistemas o aplicaciones vulnerables. Cuando se consigue acceso, brinda herramientas de post explotación que se encuentran en Meterpreter. También permite escalar privilegios, capturar teclas presionadas, volcar el *hash* de contraseñas, buscar archivos de sistema y usar terminales de comando [51].

2.2.2.3. Aircrack-ng

Aircrack-ng es un grupo de herramientas que se utilizan para el *hacking* de redes inalámbricas. Aircrack-ng utiliza varias herramientas para descubrir los objetivos que pueden ser hackeados (Airodump-ng). Airodump-ng detecta los puntos de acceso inalámbricos y los clientes que se encuentran actualmente conectados a estos. También, utiliza la información obtenida para conseguir las credenciales de estos puntos de acceso. Esta herramienta es utilizada por *hackers* de sombrero blanco que están centrados en redes inalámbricas. El conjunto de herramientas incluye ataques como el *Fluhrer Mantin Shamir* (FMS) (contraseñas codificadas en RC4), Korek (redes con seguridad de tipo *Wired Equivalent Privacy* (WEP)), ataque *Pyshkin, Tews, Weinmann* (PTW) y ataques basados en el algoritmo de cifrado WEP [41].

2.2.2.4. Better Cap

Better Cap es considerada una herramienta potente, flexible y portátil creada para realizar ataques de tipo *Man in the middle* (MITM), manipular HTTP, *Hypertext Transfer Protocol Secure* (HTTPS) y tráfico TCP en tiempo real, buscar credenciales, etc. Además, proporciona todo lo necesario para medir vulnerabilidades, fundamentalmente de texto plano [51].

2.2.3. Movimiento lateral

Esta fase se enfoca en el proceso realizado luego de obtener acceso; consolidando y expandiendo la presencia dentro del sistema objetivo. Los atacantes se mueven de equipo en equipo después de la máquina inicial para intentar acceder a información valiosa. También, se busca ganar control adicional en el objetivo sin levantar alarmas y alertas. En ataques de mucha complejidad esta fase puede llegar a durar meses [41].

2.2.4. Escalado de privilegios

La escalada de privilegios hace referencia a un tipo de ataque que se realiza sobre una vulnerabilidad para obtener más o diferentes permisos con respecto a los asignados por defecto. La escalada de

privilegios se puede distinguir en dos tipos: escalada vertical y escalada horizontal. En la fase de escalada horizontal de privilegios un usuario obtiene los derechos de acceso de otro usuario quien tiene el mismo nivel de privilegios. La escalada horizontal da la ventaja de acceder a recursos manejados por usuarios específicos sin la necesidad de obtener una mayor cantidad de privilegios. Mientras, la escalada vertical de privilegios es necesaria cuando se requiere hacer alguna acción en el sistema, como dañar o robar información importante. En este proceso el atacante aprovecha una falla en el sistema para ganar acceso a un nivel superior de privilegios (Figura 2.4) [53].

Tras la explotación de una vulnerabilidad que da acceso a un sistema; el atacante puede tener control sobre una sesión de usuario con privilegios limitados, lo que quiere decir que no puede realizar todo tipo de acciones [52].

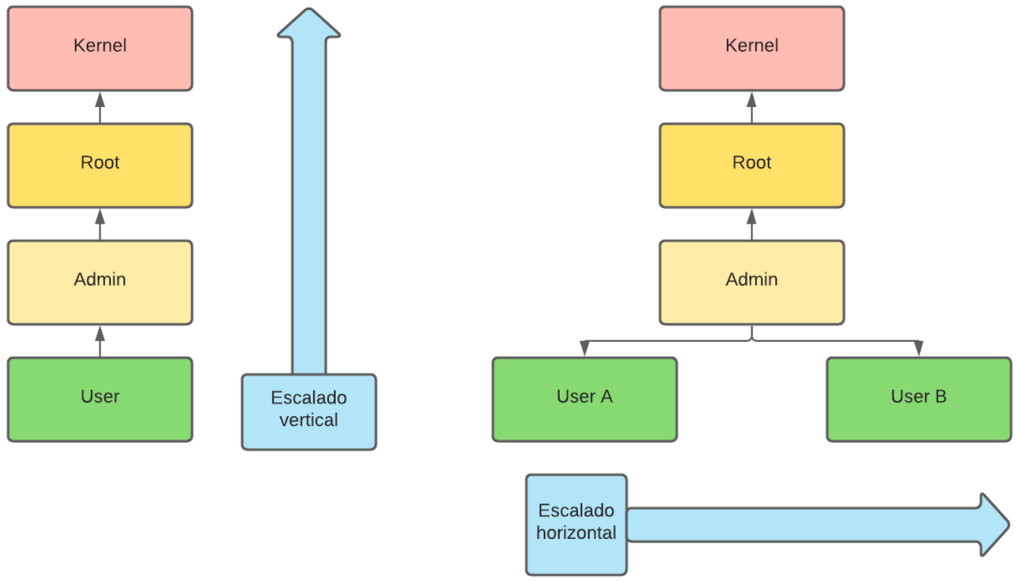


Figura 2.4: Diagrama de los tipos de escalado de privilegios (Creación propia)

La identificación de posibles *exploits* enfocados a elevar los privilegios requiere de un análisis sobre los servicios disponibles, descubiertos en la fase de reconocimiento. También, es posible utilizar módulos existentes en *Metasploit*, tales como post/multi/recon/local_exploit_seggester que presentan sugerencias respecto a posibles *exploits*, o lanzar desde Meterpreter el *script* denominado getsystem, que utiliza diversas técnicas para elevar privilegios [52].

2.2.4.1. *John the Ripper*

Esta herramienta permite la obtención de claves de seguridad disponibles en sistemas operativos Linux y Windows, se utiliza para realizar ataques con diccionarios. La herramienta encuentra las contraseñas de usuarios de bases de datos personales o de sistemas en red y aplicaciones. También, funciona listando una serie de contraseñas usadas comúnmente y luego las encripta con el mismo algoritmo. Esta herramienta también realiza una comparación entre estos resultados y los que se tienen en la base de datos para conocer si existen coincidencias. El programa encuentra contraseñas en dos pasos. En primer lugar, identifica el tipo de encriptación de la contraseña. Esta puede ser RC4, SHA o MD5 entre otros algoritmos de encriptación. En el segundo paso, el programa intenta obtener la

contraseña original mediante la comparación de la contraseña cifrada con otras en la base de datos [41].

2.2.4.2. THC Hydra

Herramienta similar a *John de Ripper*, con la diferencia que funciona online. Esta herramienta es popular entre los *crackers* ya que utiliza diccionarios y fuerza bruta para atacar a páginas de acceso. Hydra es efectiva ante bases de datos *Lightweight Directory Access Protocol* (LDAP), *Server Message Block* (SMB), *Virtual Network Computing* (VNC) y *Secure SHell* (SSH). El *cracker* brinda a la herramienta la página de acceso de cualquier de los sistemas objetivos en línea. La herramienta prueba cada posible combinación para los campos de usuario y contraseña y los guarda *offline* para conseguir la coincidencia de manera más rápida [41].

2.2.5. Concluir la misión

Sucede cuando los atacantes tiene libre movilidad dentro de la red objetivo y recolecta pruebas sobre la información que considera importante. Cuando un ataque llega a esta fase se considera exitoso. En esta fase un *hacker* de sombrero negro puede poner a la venta la información sustraída a sectores interesados, además puede implantar *malware* para crear puertas traseras y así tener acceso al sistema de forma continua. La información puede incluir secretos de ventas, nombres de usuario, contraseñas, datos sobre el personal, documentos de alta seguridad y otros tipo de datos. Después del escalado de privilegios, mediante el uso de Meterpreter *shell* (parte de Metasploit) se puede obtener pruebas sobre la información comprometida por la existencia de la vulnerabilidad explotada y de esta manera determinar el riesgo que la vulnerabilidad representa para la organización. Como paso final el *cracker* tiene que volver hacia el punto exacto antes que se inicie el ataque, borrando los rastros que puedan aparecer en un análisis forense [41].

2.2.6. Informe de auditoría

El informe de auditoría presenta cada una de las fases que se ejecutó y hasta que punto del sistema se llegó mediante los procesos realizados (archivos encontrados, mecanismos de seguridad vulnerados). Para este propósito algunas herramientas como OpenVAS, Nessus, entre otros, crean informes acerca de las tareas realizadas, lo que facilita el trabajo. Darle sentido a toda la información que ha obtenido el *hacker* ético ya sea por cuenta propia o mediante el uso de alguna herramienta es parte de las habilidades que debe tener, para así poder reportarlo en un informe de auditoría. El *hacker* ético además, debe proveer recomendaciones para mejorar la seguridad en la red, ya sea el caso de dispositivos anticuados que tengan que ser reemplazados por equipos actualizados, actualizaciones a sistemas operativos con vulnerabilidades conocidas, impartir recomendaciones al personal para evitar ataques de ingeniería social, entre otras [54].

2.3. Cadena de eliminación (*Kill chain)*

El marco metodológico *Cyber Kill Chain* es una parte de la estrategia de gestión de riesgos para la identificación y prevención de intrusiones cibernéticas. El modelo identifica las fases que realizan los *crackers* para lograr de forma exitosa el ataque. Una intrusión es una oportunidad para conocer a los

agresores y usar su perseverancia como ventaja [55]. El modelo *Cyber Kill Chain* se compone de siete fases [3]. Las etapas pertenecientes al modelo se presentan en la Figura 2.5.

Figura 2.5: Modelo *Cyber Kill Chain* [3]

1. **Reconocimiento:** En esta fase se investiga, identifica y se seleccionan los objetivos [56]. Los *crackers* planean el proceso de intrusión. Además, realizan una búsqueda de las debilidades del objetivo para un ataque exitoso. En esta fase los atacantes se concentran en recolectar correos electrónicos, identificar empleados en redes sociales, obtener comunicados de prensa, y descubrir servidores expuestos en Internet [55].
2. **Preparación:** Se planifica el ataque y se selecciona el contenido malicioso a utilizar. Para el desarrollo de *malware* se emplean herramientas automatizadas [56]. Algunos procesos a realizar en esta etapa por el *cracker* son el desarrollo de *exploits* y de archivos señuelo para presentar a la víctima, la selección del *backdoor* y la infraestructura apropiada de control para la explotación, la designación de una *Identification* (ID) de misión para el *malware*, y finalmente, la compilación del *backdoor* [55].
3. **Entrega:** Se envía el *malware* al entorno del objetivo a través de correo electrónico, interacciones en redes sociales, sitios web y medios removibles *Universal Serial Bus* (USB)[55, 56].
4. **Aprovechamiento:** Una vez enviado el *malware* hacia la víctima, la fase de explotación activa el código de intruso. Generalmente, los ataques apuntan a una vulnerabilidad de una aplicación, sistema operativo o servidores, pero también es posible explotar a los usuarios o aprovechar una característica del sistema operativo que ejecute automáticamente el código malicioso. Algunas de las formas existentes para desencadenar un *malware* son abrir un correo electrónico o hacer *click* en un enlace malicioso [56].
5. **Instalación:** En esta fase se realiza la instalación de un troyano de acceso remoto o un *backdoor* en el sistema de la víctima para explotar las vulnerabilidades existentes permitiendo al *cracker* mantener el acceso al entorno por un periodo amplio de tiempo [56]. Los atacantes en esta fase pueden realizar actividades como:

   a) Instalar un *webshell* en un servidor web.
   b) Instalar un *backdoor* en la víctima.

c) Crear un punto de acceso al agregar servicios, AutoEjecutable, etc.

d) Algunos *crackers* presentan paso a paso el archivo de ataque para presentar el *malware* como parte de la instalación del sistema operativo estándar [55].

6. **Comando y control:** En la fase de *Command & Control* (CyC) se alcanza el control remoto de la víctima a través de un servidor [57]. El *malware* abre un canal de comandos para habilitar al *cracker* a manipular de forma remota a la víctima. Para este proceso se habilita un canal de comunicación bidireccional hacia una infraestructura CyC. Los canales CyC están en la web, DNS y protocolos de correo electrónico. La infraestructura CyC puede ser propiedad del *cracker* o pertenecer a otra red de víctimas [55].
7. **Acciones sobre objetivos:** Una vez completados los procesos pertenecientes a las seis fases previas, los intrusos pueden tomar medidas para lograr sus objetivos originales. Los *crackers* que han obtenido acceso a la víctima pueden realizar acciones como recolección de credenciales de usuario, escalada de privilegios, reconocimiento interno, recolección y exfiltración de datos que involucran a la recolección, encriptación y extracción de información del entorno de la víctima, violaciones de integridad de datos o disponibilidad y destrucción de sistemas, sobreescritura, y corrupción o modificación de datos [55]. Alternativamente, los intrusos pueden únicamente desear el acceso a la víctima inicial para usarla como punto de salto para comprometer sistemas adicionales y moverse dentro de la red [56].

2.4. Pruebas de penetración

Una prueba de penetración se define como un intento legal y autorizado para ubicar y explotar de manera exitosa sistemas con el propósito de hacerlos más seguros. El proceso incluye una búsqueda de vulnerabilidades, así como proporcionar *Proof of Concept* (PoC) con el objetivo de mostrar que las vulnerabilidades son reales. Una prueba de penetración es una subclase del *Hacking* ético y siempre termina con recomendaciones específicas para abordar y arreglar los problemas descubiertos durante la prueba. En general, este proceso es utilizado para ayudar a asegurar las computadoras y redes en contra de ataques futuros.

Las pruebas de penetración ayudan a cerrar la brecha existente entre la protección del sistema de seguridad de una organización y la exposición de sus riesgos de seguridad al evaluar si los controles de seguridad son adecuados y si funcionan de forma correcta. Las pruebas de penetración pueden fortalecer los procedimientos de seguridad de una organización, así como mejorar la eficiencia de la gestión de riesgos. Además, puede aumentar el grado de transparencia mediante la evaluación de datos sensibles que pueden estar expuestos y cómo la red puede comprometerse por elementos humanos. Finalmente, el principal beneficio de las organizaciones es aprender de la experiencia de pruebas de penetración y mejorar sus sistemas de seguridad al analizar sus vulnerabilidades e implementar los cambios propuestos de manera oportuna [58].

2.4.1. Tipos

Existen cinco tipos de pruebas de penetración que se ejecutan en un proceso de *Hacking* ético: pruebas de aplicación web, aplicación móvil, ingeniería social, física y red. Actualmente, una prueba de penetración de aplicación *web* es común, ya que su aplicación abarca datos críticos como números de tarjeta de crédito, nombres de usuario y contraseñas. Las pruebas de aplicación móvil son el tipo más

nuevo de prueba debido a que las organizaciones desean asegurarse que sus aplicaciones móviles sean lo suficientemente seguras para que los usuarios confíen cuando brinden información personal al usar sus aplicaciones. En una prueba de ingeniería social se utilizan ataques de *phishing* y ataques al navegador para engañar al usuario a hacer cosas que no tenía la intención de hacer. También, en una prueba de penetración física, se pide el ingreso de forma física al edificio de la organización y la comprobación de los controles de seguridad como cerraduras y mecanismos de *Radio Frequency Identification* (RFID) [59]. En el proyecto se implementará la prueba de penetración de red.

En una prueba de red se comprueba el entorno de red para las potenciales vulnerabilidades de seguridad y amenazas. Esta prueba se divide en dos categorías: externas e internas. Una prueba de penetración externa involucra pruebas en las direcciones de IP públicas, mientras que, en una prueba interna, el *hacker* ético se puede convertir en parte de la red interna y testear la red. Para realizar esta prueba es posible que la organización proporcione un acceso VPN a la red o puede ser necesario ir físicamente al entorno de trabajo; esto depende de las reglas de compromiso definidas previamente a la ejecución de la prueba [59].

2.4.2. Categorías

Una prueba de penetración puede ser categorizada en tres tipos: caja negra, caja blanca o caja gris dependiendo de lo que la institución desea analizar y el paradigma de seguridad a utilizar.

- **Caja negra:** Una prueba de penetración de caja negra es en donde se proporciona poca o ninguna información acerca de un objetivo específico. En el caso de una prueba de penetración de red; la red perimetral *Demilitarized Zone* (DMZ) y el sistema operativo del objetivo, la versión del servidor, etc, no van a ser proporcionados; la única información brindada son los rangos IP a analizar. En el caso de una prueba de penetración de aplicaciones web; el código fuente de la aplicación web no es proporcionado. Este es un escenario común que sucede cuando se realiza una prueba de penetración externa.
- **Caja blanca:** Esta prueba es en donde casi toda la información del objetivo ha sido proporcionada. En el caso de un prueba de penetración de red, se brinda información sobre la aplicación que se está ejecutando, las versiones correspondientes, el sistema operativo, etc. En una prueba de penetración de aplicaciones web; se entrega el código fuente de la aplicación, habilitando de esta manera a realizar el "análisis del código fuente" estático/dinámico. Este escenario es común en las pruebas de penetración internas, ya que las organizaciones están preocupadas acerca de fugas de información.
- **Caja gris:** En esta prueba, un poco de información se entrega y otra se oculta. En el caso de una prueba de penetración de red, la organización proporciona los nombres de la aplicación ejecutándose detrás de una dirección IP. En una prueba de penetración de aplicaciones web, se brinda un poco de información extra como servidores *back end* y bases de datos [59].

2.4.3. Metodologías

Una prueba de penetración está basada en metodologías que ayudan a preservar su secuencia de ejecución y exponer de forma clara y concisa la información adquirida [39]. Estas metodologías permiten desarrollar la mejor estrategia con el objetivo de ejecutar una prueba de penetración exitosa [60]. Dentro de este ámbito, existen diversas metodologías utilizadas en el contexto de auditoría de

sistemas. Las metodologías principalmente utilizadas son OSSTMM, ISSAF, OWASP, y finalmente, PTES.

Considerando el análisis realizado por Álvarez [60] se presenta una descripción de la metodología PTES. Para analizar esta metodología se consideraron aspectos relevantes como: a) ámbito y enfoque; aludiendo a la clase de institución que la emplea, b) alcance; considera la cantidad de tareas que pueden ser incluidas, c) profundidad; se refiere al grado de especificación del marco metodológico, d) evaluación de riesgos; expresa el grado de severidad de un riesgo y la forma de atenuar su efecto, y finalmente; las fases pertenecientes a esta metodología para la realización de pruebas.

Meza [61] afirma que la metodología PTES facilita el desarrollo de las pruebas de intrusión pues determina el método y las herramientas necesarias en las etapas de una prueba de penetración. Es decir, la metodología emplea un lenguaje sencillo de entender con el objetivo de que el personal de dirección gerencial entienda las vulnerabilidades y los procesos de explotación realizados por el auditor; también propone las herramientas sugeridas en cada fase. Este estándar se basa en la metodología OSSTMM con la finalidad de desarrollar una auditoría considerando sus procedimientos más utilizados [60]. Al considerar ámbito y enfoque, la metodología es adaptable en todo entorno de aplicación y puede ser utilizada conjuntamente con OWASP, brindando rapidez y fundamento a la evaluación. Con respecto al alcance, PTES se encamina únicamente a niveles técnicos [60]. Con respecto a la profundidad, PTES define procedimientos y manuales correspondientes a escenarios específicos [62]. En cuanto a la evaluación del riesgo, se emplean niveles de riesgo enfocados a un lenguaje de negocios y que presente un análisis cuantitativo para permitir una comunicación clara con el cliente [60].

El estándar de ejecución de pruebas de penetración está conformado por siete etapas: antes de empezar el desarrollo, recolección de información, modelado de amenazas, análisis de vulnerabilidades, explotación, post-explotación e informe de auditoría. La Tabla 2.1 presenta las fases con su descripción [60].

2.4.4. Limitaciones

De acuerdo con la *Information Technology Association of Canada* (ITAC), no se puede esperar que las pruebas de penetración identifiquen todas las vulnerabilidades de seguridad, ni se garantiza que el proceso sea 100 % seguro. Los métodos de *hacking* pueden crear exposiciones no previstas durante las pruebas de penetración. Por lo tanto, es posible que una vez ejecutadas las pruebas de penetración se desarrollen incidentes de *hackeo* debido a que es imposible tener una protección completa, sino únicamente una buena protección para el sistema de seguridad de organización.

Las pruebas de penetración implican tomar capturas de pantalla o copiar información sensible como evidencia para probar que el sistema tiene deficiencias de seguridad. Sin embargo, hay muchas restricciones en el alcance de la información que estará disponible y accesible de forma legítima para el *hacker* ético. De esta forma, una prueba de penetración no puede simular de forma correcta las actividades de un *cracker* debido a que los atacantes tienen acceso a toda la información sin limitaciones. En primer lugar, las pruebas de penetración están regidas por leyes y obligaciones contractuales del sistema de la organización porque si la prueba recupera involuntariamente información altamente confidencial, esto resulta en la violación de leyes y el incumplimiento de acuerdos contractuales. En segundo lugar, si la información está localizada y asegurada en otro país, las leyes de ese país pueden prohibir lo que un *hacker* ético puede realizar. Finalmente, que las empresas externalicen su infraestructura informática pueden restringir técnicas similares debido a acuerdos de licencia. Todas

Tabla 2.1: Fases de la Metodología PTES

Fases	**Descripción**
Pre-compromiso	Establece el alcance de la auditoría, así como el tiempo y modelo de la prueba. En esta fase se presentan las especificaciones de ingeniería social y los permisos necesarios para el desarrollo de las pruebas de penetración.
Recolección de información	Se reúne información de la institución empleando motores de búsqueda y se establecen los objetivos.
Modelado de amenazas	Se determinan y organizan los activos primarios y secundarios.
Análisis de vulnerabilidades	Se desarrolla una búsqueda de vulnerabilidades en aplicaciones web, análisis de tráfico y ataques de fuerza bruta.
Explotación	Se evade la seguridad del sistema con el objetivo de continuar con la auditoría.
Post-explotación	Se reúne la evidencia de los ataques desarrollados y se evalúa el impacto de los riesgos encontrados. También, se efectúa el proceso de borrado de huellas y se hace permanente la invasión al sistema.
Informe de auditoría	Se elaboran informes técnicos y ejecutivos que describen los resultados obtenidos de la auditoría de seguridad.

estas restricciones implican que las organizaciones y *hackers* éticos deberían tomar medidas adicionales para reducir los riesgos de responsabilidad no deseada mediante acuerdos escritos detallados entre la empresa y el *hacker* ético para definir el alcance, objetivo, términos y cualquier limitación en el contrato.

Además, una prueba de penetración se realiza normalmente con recursos limitados durante un período de tiempo específico. Por lo tanto, una vez que un *hacker* ético ha identificado el riesgo actual y las amenazas del sistema, la organización debe tomar inmediatamente medidas correctivas para mitigar las lagunas de seguridad existentes debido a vulnerabilidades cibernéticas existentes y disminuir la exposición potencial a ataques [58].

2.5. Vulnerabilidades cibernéticas

Algunos de los tipos de vulnerabilidades más utilizadas para realizar ataques cibernéticos se describen a continuación.

2.5.1. Vulnerabilidad de inyección

La vulnerabilidad de inyección ocurre cuando un *cracker* explota un código inseguro para insertar su propio código en el programa. Debido a que el programa es incapaz de diferenciar el código insertado de su propio código, los *crackers* pueden utilizar los ataques de inyección para acceder a áreas seguras e información confidencial como si fueran usuarios reales. Algunos ejemplos de vulnerabilidades de inyección incluyen: inyecciones *Structured Query Language* (SQL), inyecciones de comando, inyecciones *Carriage Return and Line Feed* (CRLF) e inyecciones LDAP. Las pruebas de seguridad pueden revelar

las fallas de inyección existentes y sugerir técnicas para corregirlas tales como quitar caracteres especiales de la entrada del usuario o escribir consultas SQL parametrizadas [63].

La vulnerabilidad de inyección SQL es explotable a través de la inyección de códigos SQL. Estos códigos son conocidos como carga útil maliciosa en instrucciones SQL, y se dan durante el ingreso de datos a través de una página web. Durante la ejecución de una instrucción SQL, una carga maliciosa controla el servidor de base de datos de la aplicación web. Debido a que esta vulnerabilidad de inyección puede afectar un sitio web o aplicación web que hace uso de una base de datos basada en SQL. La forma más común de este ataque consiste en la inserción directa de código malicioso en parámetros concatenados con comandos SQL y la ejecución de este código.

2.5.2. Vulnerabilidad de autenticación rota

Cuando la autenticación de sesión no se implementa correctamente representa un gran riesgo de seguridad. En el caso de que los *crackers* conozcan estas vulnerabilidades, pueden suplantar la identidad de un usuario legítimo. La autenticación multifactor representa una forma eficaz de mitigar esta vulnerabilidad, así como escaneos mediante el *Dynamic application security testing* (DAST) y la *Scanner Control Application* (SCA) para detectar y eliminar errores de implementación en el código programado [63].

2.5.3. Vulnerabilidad de entidades externas XML

Este riesgo ocurre cuando los *crackers* son capaces de subir o incluir contenido *Extensible Markup Language* (XML) malicioso debido a código inseguro, integraciones o dependencias. Un escaneo SCA realizado de forma regular puede ayudar a encontrar riesgos y componentes de terceros con vulnerabilidades conocidas y alertar al usuario acerca de ellas. La probabilidad de un ataque de entidades externas XML se reduce al deshabilitar el procesamiento de entidades externas XML [63].

2.5.4. Vulnerabilidad de control de acceso

En el caso de que la identificación y la restricción de acceso no se implemente de forma correcta, es fácil para los *crackers* obtener información. Cuando existen fallas en el control de acceso, los usuarios no autorizados pueden tener acceso a archivos o incluso la configuración de privilegios de usuario. Los errores de configuración y las prácticas de control de acceso inseguro son difíciles de detectar. Las pruebas de penetración pueden detectar la falta de autenticación, pero se deben utilizar otros métodos para determinar los errores de configuración. Los controles de acceso débiles y los problemas con la gestión de credenciales se pueden prevenir con prácticas de codificación seguras, así como medidas preventivas como el uso de contraseñas para cuentas y controles administrativos así como el uso de autenticación de factores múltiples [63].

2.5.5. Vulnerabilidad XSS

La vulnerabilidad *Cross-site Scripting* (XSS) permite a los *crackers* aprovechar las *Application Programming Interfaces* (APIs) y manipular los *Document Object Models* (DOMs) para recuperar datos o enviar comandos a una aplicación. Esta debilidad amplía la superficie de ataque, lo que permite secuestrar cuentas de usuario, acceder al historial de navegación, difundir troyanos y gusanos, etc.

Esta vulnerabilidad se puede prevenir al usar la codificación de datos y la validación de entradas. Para sanitizar los datos se debe validar si es el contenido que se espera y codificar la información como una capa extra de protección [63].

2.5.6. Vulnerabilidad DROWN

La *Transport Layer Security* (TLS) es uno de los principales protocolos responsables de la seguridad del transporte de datos en Internet. TLS y la *Secure Sockets Layer* (SSL)v3 han sido el objetivo de un gran número de ataques criptográficos en la comunidad de investigación, tanto en implementaciones como el protocolo mismo. Una de las vulnerabilidades explotadas es *Decrypting RSA using Obsolete and Weakened Encryption* (DROWN). Esta severa vulnerabilidad afecta a HTTPS y otros servicios que confían en SSL y TLS. Estos protocolos permiten a cualquiera en Internet navegar en la web, usar correo electrónico, comprar en línea y enviar mensajes sin terceras partes que puedan leer la comunicación. DROWN permite a los *crackers* romper la encriptación y leer o robar información sensible. Los datos obtenidos pueden incluir nombres de usuario, contraseñas, datos de tarjetas de crédito, correos electrónicos, mensajes instantáneos y documentos confidenciales. La mayoría de los clientes TLS actuales no soportan SSLv2. Si un servidor web está configurado para usar SSLv2, y ejecuta OpenSSL es vulnerable a un ataque DROWN el cual descifra tráfico TLS. En algunos escenarios comunes, un atacante también puede suplantar un sitio web seguro e interceptar o cambiar el contenido que el usuario observa [64].

2.6. Trabajos relacionados

En un artículo redactado por Jorge Veloz, Andrea Alcivar y Carlos Silva se habla del *Hacking* ético como una metodología para descubrir fallas de seguridad en sistemas informáticos utilizando herramientas integradas en sistema operativo Kali Linux [65]. Dichas herramientas son segmentadas por la metodología, en este artículo para la fase de descubrimiento se utilizaron las herramientas Maltego y Set, para la fase de escaneo utilizaron NMAP y Armitage por último para las fases de toma y mantenimiento de acceso se utilizó Metasploit. Otras herramientas utilizadas en este artículo son Ultrasurf para navegador en Windows, y el navegador *The Onion Router* (TOR) que permite enmascarar la dirección IP del equipo [65]. Se realizó una revisión bibliográfica en los artículos [41, 45, 51] en donde se encontraron algunas herramientas disponibles para seguridad informática, tales como: parrot security, black arch y Kali Linux. Francisco Mora en su tesis de fin de máster realiza el análisis, diseño y desarrollo de una aplicación para la ejecución automática de pruebas de penetración. Donde utiliza un enfoque modular para realizar la obtención de información y la explotación de vulnerabilidades empleando herramientas existentes en Kali Linux [66].

Como cualquier otra institución, las IES tienen como prioridad proteger su información crítica. Esta información consiste en datos de enseñanza, aprendizaje, investigación científica, gestión administrativa y datos culturales [7]. Esto con el objetivo de brindar un acceso seguro a los servicios que brinda y prevenir infiltraciones por agentes externos o internos que puedan afectar a la información institucional [18]. Los ataques en las IES suelen dirigirse a la información crítica, por lo que los administradores de la red deben tener un plan de acción para salvaguardar los datos [20]. La violación de datos en una IES puede afectar gravemente sus intereses (reputación, operación y finanzas).

A nivel internacional existen trabajos de titulación tales como el de Juliana Zapata García de la

Universidad Nacional de Sabaneta Colombia [67]. Donde hace uso de pruebas de penetración para la validación de seguridad de aplicaciones web basado en el top 10 de vulnerabilidades de OWASP [63]. Además, en la tesis de Juliana Zapata se determina la importancia de la planeación en las pruebas de penetración, así como que realizar este tipo de pruebas ayuda en la determinación de vulnerabilidades y determinar el riesgo que estas suponen. Por último, determina que el top 10 de vulnerabilidades de OWASP es una guía clara de las vulnerabilidades que a día de hoy afectan a más compañías/instituciones o son de fácil explotación [67]. En el artículo escrito por M. E. Alzahrani se realiza una auditoría de ciberseguridad a la Universidad de Albaha usando una herramienta desarrollada por los autores [26]. La herramienta utilizada en el artículo crea un reporte completo acerca de la ciberseguridad de la universidad siguiendo el estándar ISO 27002. Además, provee asesoría al centro de tecnologías de información en la determinación del estado de la seguridad informática [26].

En Ecuador existen a día de hoy trabajos de titulación o artículos en los que se usa el *Hacking* ético para estimar o mejorar la seguridad informática de las universidades. Es el caso de la Universidad Nacional de Chimborazo donde mediante *Hacking* ético se realizaron mejoras en la red inalámbrica, se analizó la seguridad de las redes inalámbricas, sistemas de encriptación, mecanismos y factores de seguridad en la *Wireless Local Area Network* (WLAN). Se realizó un estudio de tipo longitudinal, lo que significa que se hicieron pruebas cada cierto periodo de tiempo para luego examinar su variación, primero sin la utilización de *Hacking* ético y luego de utilizar el manual de políticas de seguridad generado. De esta manera se estima el porcentaje de utilidad que se obtiene al aplicar medidas de seguridad a la red inalámbrica. De acuerdo a la auditoría realizada y el manual de políticas concluye que se podría mejorar la seguridad en al menos un 25 % [29]. También Manuel Jaya de la Universidad Politécnica Salesiana de Quito realizó un estudio de la seguridad informática en el colegio católico José Engling mediante *Hacking* ético. En este caso específico se utilizó la metodología ISSAF que está diseñada para evaluar la red, sistemas y aplicaciones. Mediante la auditoría realizada se obtuvo que la institución educativa es vulnerable en un 66,7 % frente a ataques a la confidencialidad de información, 50 % contra la integridad y un 50 % a la disponibilidad [68].

2.7. Conclusiones

Mediante la investigación realizada en torno a la seguridad informática en las IES, se pudo determinar la importancia que posee el *Hacking* ético en este campo debido a la cantidad de información que se maneja, ya que la filtración de información afectaría gravemente a los intereses y reputación de una institución. En Ecuador se han realizado auditorías de *Hacking* ético [29] [68] en diversas instituciones y los resultados obtenidos determinan la calidad de los sistemas de seguridad informática existentes. Existe documentación [29, 39, 60, 69] donde se realiza una auditoría de seguridad informática; de dichos documentos se ha verificado la existencia de varios métodos para el desarrollo de pruebas de penetración que pueden ser utilizados bajo condiciones específicas. Sin embargo, generalmente se emplea la metodología PTES.

Se pudo verificar la existencia de variedad de herramientas utilizadas por *hackers* éticos, *pentesters* y *Red* y *Blue teams*. Las herramientas analizadas tienen como base al sistema operativo Kali Linux ya que es la plataforma preferida por los usuarios debido a la versatilidad que ofrece así como por las herramientas preinstaladas. Una vez que se ha alcanzado la fase de ejecución de un ataque ya no depende de la herramienta que se utiliza, sino de la experiencia que el *hacker* ético tenga en el campo,

caso contrario se tendrá que obtener la mayor cantidad de conocimiento acerca de la empresa que está atacando y la vulnerabilidad en específico que se está explotando. Con la investigación teórica tanto de *Hacking* ético, pruebas de penetración, *Red* y *Blue team* y las metodologías existentes de pruebas de penetración, se determina las fases a utilizar en el diseño e implementación del *framework*. Así como las herramientas y procesos disponibles, para posteriormente evaluar su funcionamiento y compatibilidad.

A partir de la información obtenida sobre pruebas de penetración, se determina las limitaciones que tienen las mismas. Estas pruebas no garantizan un proceso 100 % seguro, ya que se pueden producir peligros no previstos durante su realización. Finalmente, una vez terminada la prueba de penetración, la empresa/institución debe tomar las medidas correctivas correspondientes para mitigar los vacíos existentes en la seguridad informática.

Capítulo 3

Diseño e implementación

Para la elaboración de este trabajo de titulación fue necesaria la utilización de un conjunto de herramientas y librerías. En este capítulo se presenta el análisis sobre el diseño del *framework* así como su implementación. El capítulo comienza realizando un análisis de las herramientas (Sección 3.1). A continuación, se examina la arquitectura de la solución (Sección 3.2) y finalmente se presenta la implementación del *framework* (Sección 3.3).

3.1. Análisis de herramientas

Para realizar un proceso de pruebas de penetración exitoso se pueden usar una variedad de herramientas. Cada una de las herramientas analizadas son útiles en diferentes fases del *Hacking* ético pero solo algunas se pueden implementar en una herramienta automatizada. Este proceso se consigue comprobando la funcionalidad o mediante otros estudios realizados por terceros.

En la fase de recopilación de información se busca la mayor cantidad de información posible sobre el objetivo. Esta información obtenida va desde información que puede ser encontrada en la red sin interactuar con el objetivo hasta enviar peticiones o paquetes al objetivo y analizar su respuesta. Razón por la cual, la fase se divide en 3 subfases.

3.1.1. Recopilación pasiva de información

La primera subfase trata de conseguir todos los datos o información acerca de la infraestructura tecnológica del objetivo. Este proceso facilita las siguientes fases del proceso. Este análisis se consigue sin que las actividades realizadas por el *pentester* sobre dicho objetivo tengan interacción directa. Esta subfase resulta difícil de realizar, debido a que se tiene que utilizar diversos métodos entre ellos, búsqueda manual en bases de datos, y a menudo proporciona datos que son poco concluyentes. La manera habitual de la recolección pasiva de información es mediante el acceso a información almacenada en lugares públicos.

Una herramienta muy utilizada para recopilar información de forma pasiva es Shodan, que consiste

en un buscador que no indexa las páginas web sino recorre Internet y busca todas las direcciones que se encuentran conectadas. También, analiza los puertos que tienen estos sistemas mediante peticiones, e indexa esta información para posteriormente presentarla al usuario. Otra herramienta utilizada en esta subfase lleva el nombre de *The-Harvester* que Kali Linux trae instalada por defecto. Esta herramienta permite la utilización de varios motores de búsqueda tales como google, trello, linkedin, shodan, etc. A su vez, esta herramienta crea un archivo de reporte usando formatos conocidos (XML y *JavaScript Object Notation* (JSON)). Dicha característica la hace útil al momento de hacer uso de la información en fases posteriores o cuando se realice un informe de auditoría. NMAP es una herramienta útil y versátil que permite el escaneo pasivo de *hosts* en la red interna mediante la utilización del comando "−sn" que no realiza un escaneo de puertos sino que muestra a los *hosts* que respondieron al escáner; es decir, los *hosts* que se encuentran activos en la red. Adicional a esto NMAP permite generar un informe de resultados utilizando un formato XML. *Wireshark* puede ser utilizada para realizar la obtención pasiva de información, debido a que se sitúa en la red y se encarga de visualizar el tráfico existente en la red interna, lo que puede dar información útil acerca de los *hosts* activos en la red, con la desventaja para el caso de estudio, de funcionar sólo bajo una interfaz gráfica manual lo cual no posibilita la automatización.

3.1.2. Recopilación activa de información

Esta subfase trata de realizar la recolección de información sobre el objetivo determinado utilizando métodos que interactúan de manera directa con la organización, usualmente mediante el envío de tráfico de red. Dentro del alcance se encuentran actividades como: escáneres de puertos, servicios y de *hosts*. En ocasiones este tipo de tareas mediante estas técnicas suele ser detectada como actividad sospechosa o maliciosa.

La herramienta más utilizada para el escaneo de puertos, servicios y *hosts* lleva el nombre de NMAP. Como se ha visto en secciones anteriores, NMAP es una herramienta potente a la hora de realizar un proceso de *Hacking* ético, debido a la cantidad de funcionalidades y soporte que posee. La herramienta determina el estado de los puertos del objetivo, tales estados pueden ser: abierto si el puerto se encuentra expuesto a la red, cerrado si el puerto no responde peticiones, filtrado si el puerto en cuestión se encuentra tras un *firewall* y si no se puede determinar el estado entre abierto o filtrado, se presenta abierto/filtrado. NMAP provee las herramientas para escaneo de puertos más completa mediante argumentos ingresados por la terminal. Los comandos para los propósitos actuales son: "−sS" para escaneo de puertos TCP, "−sU" para escaneo de puertos de UDP, y "−Pn" para descubrimiento de puertos sin la utilización de paquetes ICMP.

3.1.3. Identificación de vulnerabilidades

Esta fase consiste en la identificación de fallos de seguridad que se encuentran presentes en los sistemas que se están evaluando. Este tipo de fallos abarca desde errores en la configuración de un servicio hasta vulnerabilidades en determinados servicios que sean publicados y puedan comprometer la integridad del mismo. En esta fase se han encontrado variedad de herramientas que facilitan el proceso. La herramienta más utilizada y conocida de estas herramientas es Nessus. Este es un sistema completo para el descubrimiento de vulnerabilidades, ya que permite crear búsquedas personalizadas de vulnerabilidades así como programar su realización cada cierto tiempo, entre muchas otras funcionalidades. Hasta la versión 5, Nessus podía efectuar análisis personalizados mediante una clave de aplicación; sin

embargo, desde la versión 7 no es posible con la versión gratuita. Debido a esto, actualmente no es posible la generación de análisis y resultados desde lenguajes de programación [70].

Por otro lado, OpenVas es manejado por Greenbone y es el sucesor de código abierto de Nessus. Esta herramienta no está presente por defecto en Kali Linux, y funciona como Nessus, es decir, con interfaz de usuario. Debido a esto, no resulta de utilidad a la hora de automatizar el proceso. Otra herramienta bastante conocida al momento de realizar el descubrimiento o análisis de vulnerabilidades es Nikto. Esta herramienta permite realizar un escaneo de vulnerabilidades de tipo web. Sin embargo, debido a que la herramienta requiere interacción del usuario, no fue utilizada para el diseño del *framework*. Nuclei por otra parte es una herramienta de código abierto que permite al usuario y a colaboradores, crear plantillas personalizadas para buscar vulnerabilidades específicas y conseguir un sistema con cero falsos positivos.

3.1.4. Explotación

Luego de la fase de recopilación de información, se tiene datos acerca de las vulnerabilidades existentes en el sistema objetivo. A continuación, para medir el nivel de riesgo que puede tener una vulnerabilidad, existe la posibilidad de aprovechar la debilidad para obtener un beneficio o usar una base de datos y catalogarla. Para el proceso de explotación de una vulnerabilidad se pueden utilizar herramientas semi-automatizadas, tal es el caso de Metasploit. Esta herramienta tiene un número considerable de *exploits* y *payloads* para numerosas vulnerabilidades, lo cual lo hace una potente herramienta para explotación de debilidades informáticas, no obstante, la herramienta necesita de interacción con el usuario, por lo que para poder automatizarla, se tendría que crear módulos específicos con órdenes concretas para la explotación de cada vulnerabilidad. Armitage es una interfaz gráfica de Metasploit que permite mejorar su gestión. Aircrack-ng, es una herramienta que permite aprovechar vulnerabilidades en redes inalámbricas, quebrantando los sistemas seguridad mediante diversos métodos, debido al objetivo de este trabajo de titulación esta herramienta se desestima. Bettercap permite probar y explotar vulnerabilidades de tipo MITM. El inconveniente de esta herramienta es la interacción con el usuario y la necesidad de utilizar programas de terceros para poder ejecutar el ataque correctamente.

Al concluir la fase explotación según el acuerdo que se tenga con la empresa a analizar, se puede continuar con los procesos de escalado de privilegios y movimiento lateral. En estos procesos se hace uso de herramientas tales como *Jhon the Reaper* y THC Hydra que sirven para volcado de los *hash* de claves de usuario, este proceso se realiza de manera manual por el *hacker* ético debido a que las herramientas necesitan de mucha interacción con el usuario. En el caso del estudio actual, el proceso de explotación se realizó con dos ataques reconocidos en el análisis a la máquina virtual vulnerable. Debido a que el objetivo del *framework* es realizar un análisis automático no se implementan las fases de escalado de privilegios y movimiento lateral.

Fase	Subfase	Modulos	Herramienta	Formato de salida
1	1	theharve.py upnmap.py scaner_de_red.py	The-Harvester NMAP Código en python	xml,json xml txt
1	2	nmaptcp.py nmapudp.py nmappn.py	NMAP	xml
1	3	vunlnmaptcp.py vulners_nmap_serv.py nuclei.py	NMAP, vulners Nuclei	xml json
2	1	cve-2015-3306.py cve-2018-15473.py	Códigos en python de terceras personas.	txt

Tabla 3.1: Herramientas que se utilizaran en el *framework* en los diferentes módulos que lo componen.

3.2. Arquitectura de la solución

El *framework* implementado se llama Daniz. Daniz está desarrollado en el lenguaje de programación python y se compone de tres módulos principales que emulan a algunas fases del *Red team*: recopilación de información, explotación e informe de auditoría. Al considerar la fase de informe de auditoría es importante resaltar que el *framework* únicamente presenta un reporte con los resultados de la herramienta. Este reporte debe ser analizado por el operador del *software* para construir el informe de auditoría. En la Figura 3.1 se visualizan los módulos implementados y sus correspondientes etapas. El *framework* desarrollado se ejecuta mediante el módulo principal daniz-auto.py. Este módulo invoca a otros módulos que ejecutan las fases de recopilación de información, explotación y reporte. Los módulos generales sirven de apoyo a los módulos de las fases y subfases.

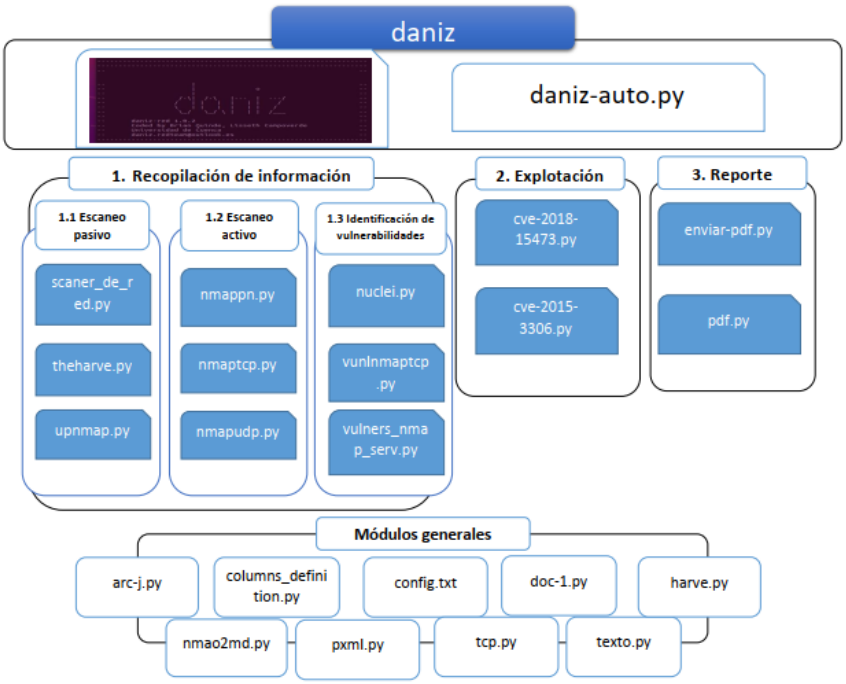


Figura 3.1: Componentes fundamentales de la Daniz

En la Figura 3.2 se presenta el diagrama de flujo del *framework* desarrollado. La herramienta se encuentra dentro de un contenedor Docker, en donde se instalan las herramientas y librerías necesarias para su funcionamiento. Para ejecutar un análisis periódico de la red de la universidad se utiliza la herramienta Cron. La frecuencia de ejecución del *framework* se configura mediante el archivo crontab .

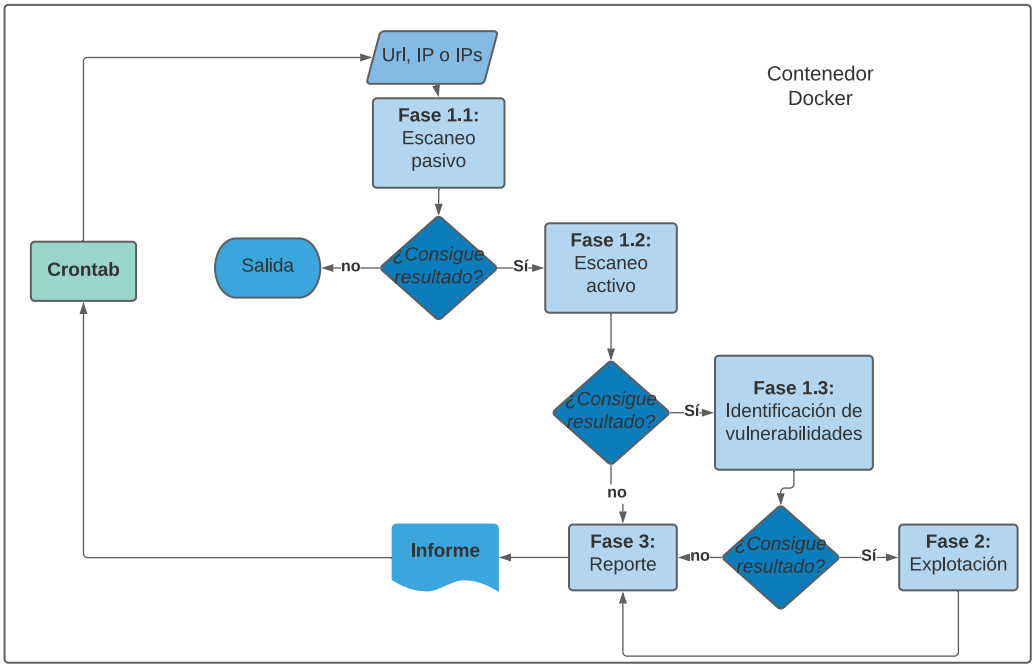


Figura 3.2: Diagrama de flujo de la aplicación daniz

El *framework* se compone de módulos y su salida es procesada por un programa principal. El programa principal como primer paso recibe el objetivo al que se va a realizar el análisis completo. Esta información se ingresa al sistema como un archivo de configuración conformado por 3 campos uno en cada línea los cuales son correo del usuario, interfaz conectada a la red y objetivo o varios separados por comas. (Listado 3.1).

```
Entrada: Archivo de configuración
Salida: Ips o URLs, correo e interfaz a utilizar.
Config =leer el archivo de configuración y dividirlo por línea.
si la lista generada supera o iguala a 2 campos
    correo = primer campo de Config
    escribir este correo en el archivo correo.txt
    interfaz= segundo campo de la lista de configuración
    intentar:
        si objetivo se puede separar por comas
            crear una lista de objetivos
        caso contrario
```

```
            objetivo = tercer campo del archivo de configuración
    excepción:
        objetivo = vacío
Salida: lista de objetivos u objetivo, interfaz
```

Listado 3.1: Algoritmo para el procesamiento del archivo de configuración

Una vez obtenidos los datos del archivo de configuración, el programa principal busca los programas ubicados en el *path* de la fase 1.1 (recopilación pasiva de información). Estos módulos encontrados se ejecutan sobre el objetivo, con el propósito de encontrar información. Este proceso de búsqueda de los módulos existentes por fase de ejecución se repetirá en cada etapa. Cada módulo genera un archivo de salida; estos archivos de salida son procesados en búsqueda de direcciones IP que sirvan de ingreso para la fase 1.2, Listado 3.2 (Recopilación activa de información).

```
Entrada: Archivos de salida de fase 1.1
Salida: Objetivos útiles para fase 1.2 de ejecución.
dirección= Directorio salida fase 1.1
lista=obtención de archivo dentro de dirección
recorrer lista
    lista texto=abrir el documento y segmentarlo por separador de línea
    lista ip=vacío
    comprobar si existe ips.
        si la ip no se encuentra en lista ip
            agregar la ip a lista ip
    salida: lista ip
```

Listado 3.2: Pseudocódigo el procesamiento de la salida de la fase 1.1

Con las IPs encontradas se prosigue con la fase 1.2. Esta fase comienza buscando los módulos existentes dentro del *path* correspondiente a la fase 1.2. Luego de este proceso se utiliza cada IP para cada uno de los módulos existentes para obtener una respuesta de cada módulo, como en la fase anterior cada módulo genera un archivo de salida propio. Estos archivos son procesados en búsqueda de direcciones IP y de puertos TCP/UDP abiertos para segmentar la búsqueda hacia objetivos más probables, listado 3.3.

```
Entrada: Archivos de salida de fase 1.2
Salida: Objetivos útiles para fase 1.3.
dirección= Directorio salida fase 1.2
lista=obtención de archivo dentro de dirección
recorrer lista
    xml=transforma el archivo de xml a diccionario
    recorre el diccionario
    si la llave es hosthint (parámetro de NMAP)
        recorrer el subdiccionario hasta encontrar host
            a=dirección ip del host
            b=Dirección física o MAC

        recorrer diccionario hasta encontrar ports
            port= sublista de puertos
            agregar [a, b] a la lista de direcciones
            agregar sublista port a la lista de puertos
    salida: lista de direcciones con puertos
```

Listado 3.3: Algoritmo para procesamiento de documentos de fase 1.2

Una vez obtenida la lista de direcciones IP que poseen puertos abiertos se puede proseguir con la fase 1.3. Igual que en fases anteriores, se busca los módulos disponibles para esta fase y se los ejecuta sobre cada una de las IPs conseguidas. A partir de los resultados sobre las vulnerabilidades existentes, los archivos JSON de Nuclei se procesan de manera que se pueda obtener una vulnerabilidad (Código *Common Vulnerabilities and Exposures* (CVE)) coincidente con las vulnerabilidades existentes en el sistema, listado 3.4. Cabe recalcar que Nuclei proporciona un archivo JSON sin un formato correcto, ya que un JSON posee un diccionario en una sola linea, mientras el archivo generado por Nuclei presenta mas de un diccionario, uno por linea, por lo que hay que usar más procesos para utilizarlo.

```
Entrada: Archivos de salida de fase 1.3
Salida: Explotación de CVEs asociadas.
dirección= Directorio salida fase 1.3
lista=obtención de archivo dentro de dirección
recorrer lista
    si el archivo es json
        a=abrir el archivo json y separarlo por el salto de línea
        recorrer lista a
            archiv= crear un documento vacío para ingresar la información obtenida.
            crear un nuevo json por cada línea leída

        num= longitud de a
        para r recorre desde 0 a num
            abrir el archivo r y convertirlo en diccionario
            c=referencia a la vulnerabilidad (Que contenga CVE)
            agregar c a la lista de CVEs.

        cveprogramas= programas para explotación cargados
        recorrer la lista cveprogramas
            recorrer la lista de CVEs
                si el elemento en cveprogramas es igual al elemento en CVEs
                    ejecuta el programa de explotación sobre la IP asociada
```

Listado 3.4: Algoritmo para procesamiento de documentos de fase 1.3

En el listado 3.4 se presenta el pseudocódigo para la ejecución del ataque asociado a la vulnerabilidad. En el caso de que las vulnerabilidades encontradas (código CVE) correspondan a los módulos de explotación existentes se ejecuta el ataque correspondiente y se genera un archivo con la información de la explotación. Los ataques realizados se basaron en los resultados obtenidos de la realización de pruebas de penetración hacia una máquina virtual vulnerable. Los módulos de ataque existentes son programas escritos con python y generados por terceras personas, con el fin de prescindir de herramientas automatizadas como Metasploit. Una vez alcanzado este punto el análisis se considera terminado y se procede con la generación de informe.

Los resultados obtenidos en cada fase y subfase del *framework* son procesados para generar un reporte final. Este informe se envía mediante email al correo ingresado en el archivo de configuración. El código de programación de la herramienta no se incluye en este trabajo de titulación debido a su extensión. El repositorio se encuentra en la plataforma GitHub, que se puede acceder mediante el enlace: `https://github.com/fabianastudillo/daniz-red.git`.

3.3. Implementación

Para el desarrollo del *framework* se ha utilizado la imagen de Docker "Kali-linux/Kali-rolling" basada en Kali debido a que esta distribución está diseñada para la auditoría y seguridad informática. El *framework* fue creado en una imagen de Docker debido a la versatilidad de instalación que Docker presenta. Para el funcionamiento del proyecto se han añadido algunos elementos necesarios a la imagen "Kali-linux/Kali-rolling" tales como las herramientas: the harvester, nuclei, NMAP y las librerías de python: mdpdf, paramiko, argparse, json, xmltodict, fpdf, email.

El lenguaje de programación utilizado es python, debido a que es un lenguaje poderoso y que ayuda al desarrollo de *software*. Con el objetivo de validar el funcionamiento de la herramienta se utilizó el entorno virtual Metasploitable que permite probar y mejorar las técnicas de *hacking* debido a que no se debe realizar un proceso de pruebas de penetración de forma arbitraria por posibles implicaciones legales. Finalmente, se usa Cron para programar la ejecución del *framework* en un tiempo especificado en el archivo de configuración. A continuación, se presentan algunas características del entorno de desarrollo así como una descripción de las herramientas mencionadas previamente.

En la Figura 3.3 se visualiza la arquitectura de *software* utilizada en el proyecto. El módulo principal generado en python recibe un archivo de configuración de texto simple y utiliza las herramientas de software The harvester, NMAP, Nuclei y python para la recolección de información y explotación de vulnerabilidades. La información obtenida se presenta en forma de un reporte generado mediante python. Para la comprobación del funcionamiento del *framework* se utilizó la máquina virtual vulnerable Metasploitable3.

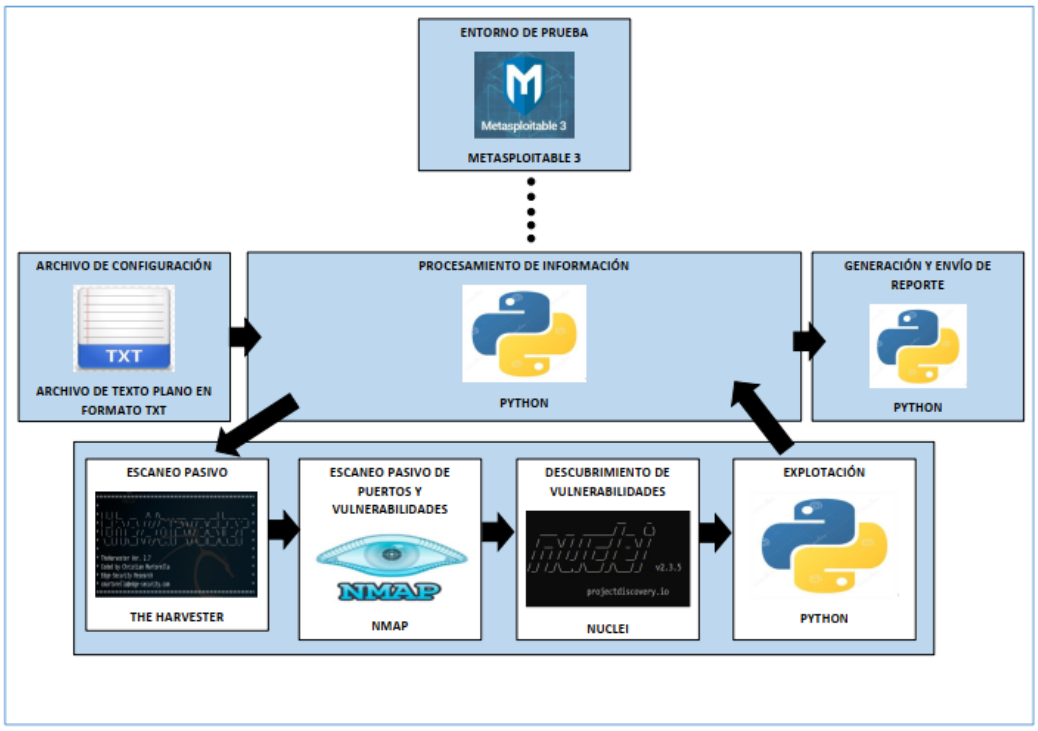


Figura 3.3: Arquitectura de *software* del *framework*

3.3.1. Sistema base

3.3.1.1. Docker

Docker [71] es una plataforma de código abierto para el desarrollo, envío y ejecución de aplicaciones. Esta plataforma permite separar las aplicaciones de la infraestructura para poder instalar *software* de forma rápida. Esto sucede debido a que Docker empaqueta el *software* en contenedores que incluyen lo necesario para que el *software* se instale y ejecute correctamente.

Al utilizar Docker es posible ejecutar varios contenedores de forma simultánea en un *host*. Los contenedores son ligeros y contienen todo lo necesario para ejecutar la aplicación, por lo que no necesita depender de lo que está instalado actualmente en el *host*. Docker proporciona las herramientas y una plataforma para gestionar el ciclo de vida de sus contenedores. Para el uso de Docker en una aplicación se siguen los pasos:

- Desarrollar la aplicación y sus componentes de soporte utilizando contenedores.
- El contenedor se convierte en la unidad para distribuir y probar la aplicación.
- Implementar la aplicación en el entorno requerido, como un contenedor [72].

3.3.1.2. Kali Linux

Kali Linux [73] es una distribución de Linux de código abierto destinada a la realización de pruebas avanzadas de penetración y auditoría de seguridad. Kali contiene varias herramientas utilizadas para pruebas de penetración, investigación de seguridad, informática forense, incluyendo escáneres de puertos como NMAP, *sniffers* como Wireshark, *suites* de ataque a redes inalámbricas como *Aircrackng*, *suites* para creación de troyanos y exploits como Metasploit o programas para descubrir claves como *John the Ripper*. Algunas de las características de Kali son:

- Incluye más de 600 herramientas para pruebas de penetración
- La distribución Kali es completamente personalizable
- Está desarrollada en un entorno seguro
- El kernel se actualiza continuamente con parches de inyección
- Los paquetes y repositorios están firmados por *GNU Privacy Guard* (GPG) [74].

3.3.1.3. Python

Python es un lenguaje de programación orientado a objetos de alto nivel. Este lenguaje de alto nivel está construido mediante estructuras de datos, combinado con escritura dinámica. Estas características lo hacen atractivo para el desarrollo de aplicaciones, así como para su uso como un lenguaje de secuencias de comandos o para conectar módulos existentes.

La sintaxis de python es simple y fácil de aprender y enfatiza la legibilidad, por lo tanto reduce el costo computacional del programa. Python soporta módulos y paquetes, lo que fomenta la reutilización del código. El intérprete de python y su biblioteca estándar están disponibles en forma de código fuente o binario para todas las plataformas principales y se pueden distribuir libremente [75].

3.3.1.4. Cron

Cron es una herramienta de Unix, Solaris y Linux que permite ejecutar tareas de forma automática en segundo plano y en intervalos regulares. Cron utiliza un archivo llamado crontab con el horario de

las entradas cron a ser ejecutadas y los tiempos específicos [4].

El archivo crontab tiene cinco campos para determinar el día, fecha y hora seguidos por el comando a ejecutar en ese intervalo (véase la Figura 3.4). El * en los valores de los campos significa que todos los valores posibles se han establecido. Es decir, si se colocan * en todos los campos el comando se ejecutará en todo momento.

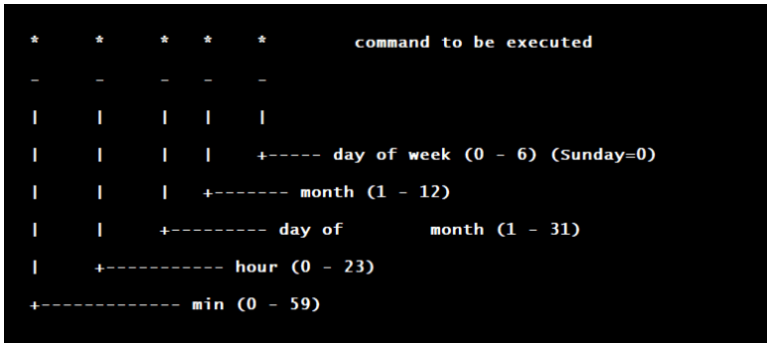


Figura 3.4: Archivo crontab [4]

3.3.1.5. Metasploitable

Se trata de una máquina virtual preparada por los creadores de Metasploit que cubre la necesidad de tener un equipo vulnerable en un entorno controlado con el propósito de realizar pruebas de seguridad informática [76]. Esta máquina virtual es utilizada por personas en la industria de seguridad informática por diferentes razones: para la explotación de redes, desarrollo de *exploits*, pruebas de *software*, etc [77].

3.3.2. Obtención de información

Como se visualizó en secciones anteriores esta fase se puede dividir en 3 subfases. Para la implementación de cada subfase se crean módulos que realizan un proceso similar pero buscan más resultados.

3.3.2.1. Recolección pasiva de información

Del proceso del análisis de herramientas se seleccionaron dos para la implementación del *framework*. The-Harvester es una herramienta que utiliza argumentos para configurar el tipo de obtención de información y el objetivo a analizar; es por esto que esta herramienta fue seleccionada para la implementación. Además, python integra la librería "os" que permite acceder a funcionalidades dependientes del sistema operativo, por lo que representa un factor fundamental para el desarrollo del *framework*.

La herramienta The-Harvester integra varios motores de búsqueda con el objetivo de buscar información, pero con el propósito de encontrar mayor cantidad de información se utilizó 4 motores de búsqueda, estos fueron: google, urlscan, linkedin, trello. El tiempo de búsqueda de la herramienta suele incrementarse por la cantidad de resultados que captura, el número predeterminado de resultados es de 500, por lo cual se limitó a 300. Esta herramienta permite crear un archivo con los resultados en formato XML y JSON lo cual facilita su tratamiento a la hora de crear un informe. También, otra herramienta que se seleccionó es NMAP ya que para un análisis de red interna se puede hacer uso

de esta, como se mencionó en apartados anteriores. Además del comando se emplea una plantilla de estilos:

```
--stylesheet=‘‘https://svn.nmap.org/nmap/docs/nmap.xsl’’.
```

Esta plantilla es útil a la hora de visualizar la información presentada por NMAP. Dicha plantilla se utiliza a partir de este punto en todos los comandos de NMAP.

3.3.2.2. Recolección activa de información

En esta subfase se utiliza NMAP usando comandos específicos de la herramienta. La razón por la cual se creó cada comando en un módulo distinto, es la carga computacional. Al separar los comandos en módulos independientes se da opción al usuario de seleccionar que módulos no utilizar moviéndolos de su carpeta original y de esta manera agilizar el análisis, no obstante el hecho de utilizar una menor cantidad de módulos puede disminuir la cantidad de resultados obtenidos. En las Figuras 3.5, 3.6, 3.7, se presenta la ejecución de los comandos presentados en este apartado. Al analizar las figuras se puede observar que el escaneo de puertos UDP es el que consume una mayor cantidad de tiempo. Todos estos análisis de pruebas se han realizado en una máquina virtual vulnerable “Metasploitable”.

Figura 3.5: Escaneo de puertos sin la utilización de ICMP

De las pruebas realizadas se obtiene el tiempo de ejecución y resultados de los diferentes comandos utilizados en NMAP.

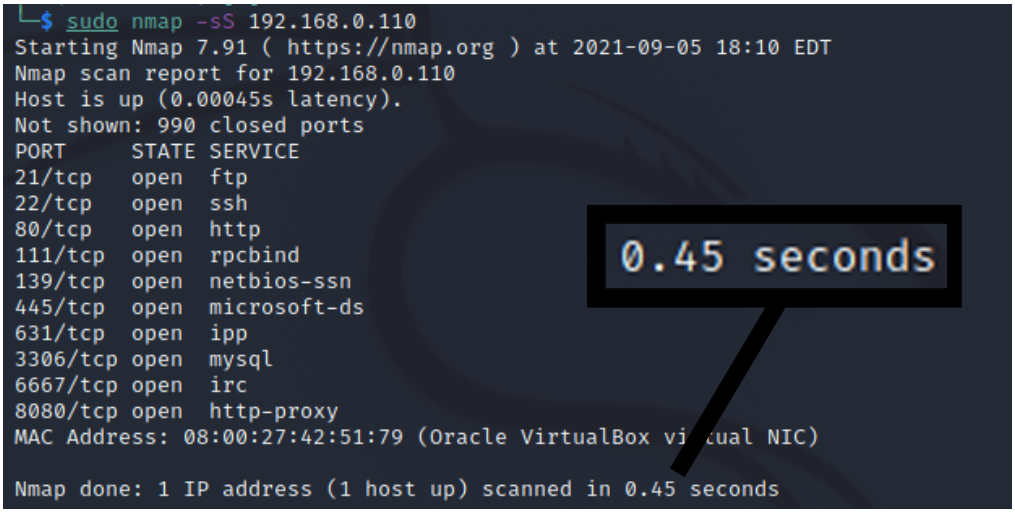


Figura 3.6: Escaneo de puertos TCP mediante peticiones

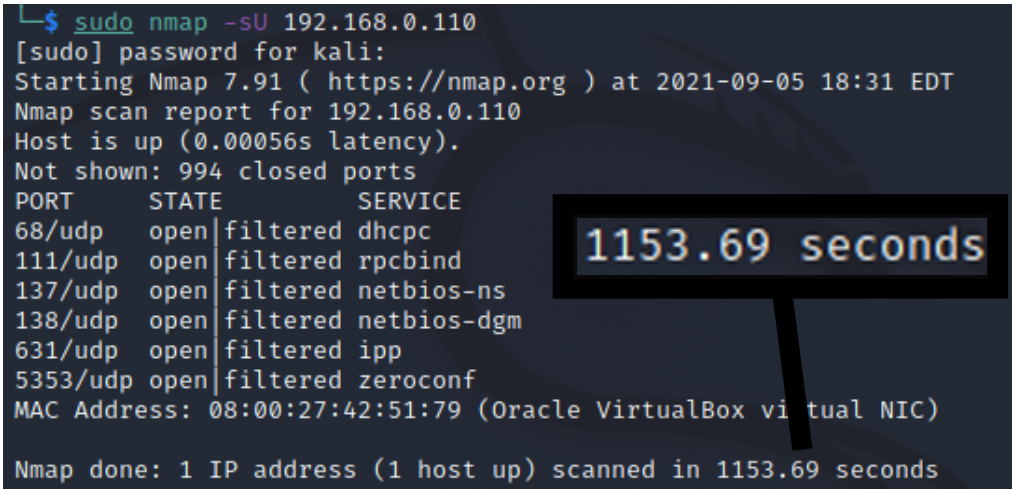


Figura 3.7: Escaneo de puertos UDP mediante peticiones

3.3.2.3. Identificación de vulnerabilidades

En esta fase se seleccionaron las herramientas Nuclei y NMAP, como se ve en la Tabla 3.1. NMAP se utilizó mediante comandos conocidos como *scripts*, uno de los *scripts* que se usa lleva el nombre de "vuln" que consigue rastrear las vulnerabilidades más conocidas en el objetivo utilizando como base los puertos TCP abiertos para su análisis [78]. El otro *script* utilizado llamado "vulners" hace uso de la base de datos de Vulners para obtener las vulnerabilidades comunes correspondientes a cada servicio descubierto durante al análisis. Este *script* posee la opción de utilizar un *Common Vulnerability Scoring System* (CVSS) para obtener las mejores coincidencias bajo cierto margen de factibilidad del 1 al 10. Sin embargo, en la versión 7.91 de NMAP que se está utilizando actualmente, esta funcionalidad no se ejecuta correctamente por lo que se tuvo que revisar los archivos de NMAP y solucionar un problema en el archivo que ejecuta el *script*. Por tanto, se realizó el cambio que se ve en la Figura 3.8. Después de la reparación en el archivo "vulners", para el caso actual, se utilizó una base mínima de 7.5 de CVSS.

```
-- NOTE[gmedian]: exploits seem to have cvss == 0, so print them anyway
if (mincvss <= v.cvss) then
  setmetatable(v, cve_meta)
  output[#output+1] = v
 end
end
```

Figura 3.8: Cambio en el archivo vulners.nse de NMAP

Por último, en la implementación de esta fase se usó la herramienta Nuclei que permite obtener vulnerabilidades utilizando plantillas *Yet Another Markup Language* (YAML) creadas por otros *hackers* éticos y validadas por el equipo de desarrollo. Además, Nuclei posee la característica de utilizar *tags* para buscar vulnerabilidades específicas. Para el desarrollo del *framework* se realizó un grupo de pruebas sobre la máquina virtual vulnerable para determinar de que forma se obtenía la mayor cantidad de resultados. En la Figura 3.9 se presenta la realización de una prueba en búsqueda de vulnerabilidades utilizando el *tag*: "CVE" que trata de encontrar vulnerabilidades conocidas (Código CVE), durante esta prueba se encontraron 2 resultados. También, en la Figura 3.10 se desarrolla una prueba con el *tag*: "exposure" que pretende encontrar vulnerabilidades en función de los servicios que utiliza el objetivo, donde se obtuvieron 2 resultados. Finalmente, en la Figura 3.11 se realiza una prueba sin la utilización de *tags* que realiza un análisis generalizado de las posibles vulnerabilidades en el objetivo, de donde se obtienen 3 resultados.

```
└─$ nuclei -u 192.168.0.110 --tags=cve

                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   2.4.3

                projectdiscovery.io

[INF] Your current nuclei-templates v8.4.5 are outdated. Latest is v8.5.0
[INF] Downloading latest release...
[INF] Successfully updated nuclei-templates (v8.5.0). GoodLuck!
[INF] Using Nuclei Engine 2.4.3 (outdated)
[INF] Using Nuclei Templates 8.5.0 (latest)
[INF] Using Interactsh Server https://interact.sh
[INF] Templates loaded: 662 (New: 158)
[INF] Templates clustered: 4 (Reduced 2 HTTP Requests)
[2021-09-05 22:08:35] [CVE-2015-3306] [network] [high] 192.168.0.110:21
[2021-09-05 22:08:37] [CVE-2018-15473] [network] [low] 192.168.0.110:22
```

Figura 3.9: Prueba con la máquina virtual utilizando el *tag* "cve "

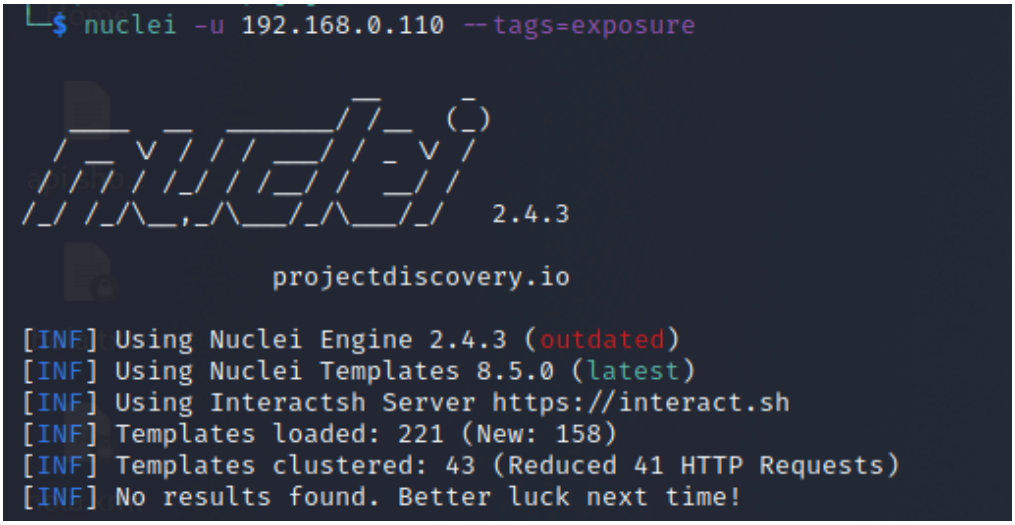


Figura 3.10: Prueba con la máquina virtual utilizando el *tag* "exposure "

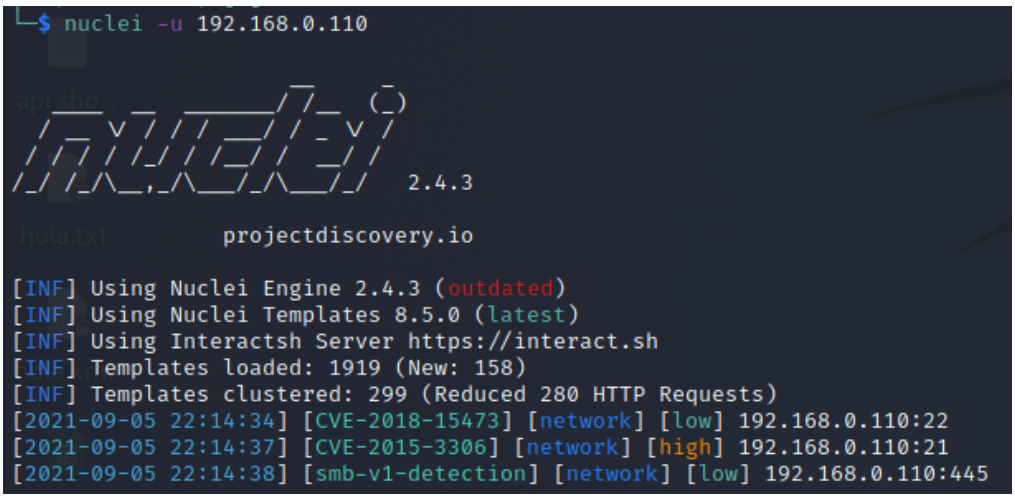


Figura 3.11: Prueba con la máquina virtual sin utilizar tags

Luego de las pruebas realizadas sobre Nuclei se determinó que la mejor manera para usarlo actualmente es sin la utilización de *tags*, es decir; de manera predeterminada ya que se usa una gran variedad de plantillas.

3.3.3. Explotación

Al momento de realizar explotación de las vulnerabilidades existentes, se pueden utilizar varios métodos. Para este caso se utilizó la explotación manual. Luego de haber determinado las vulnerabilidades existentes en el sistema objetivo se buscan en varias bases de datos tales como "https://cve.mitre.org/ ", "https://www.exploit-db.com/search", etc. Esta búsqueda se hace con el objetivo de encontrar más información acerca de las vulnerabilidades encontradas utilizando su código CVE (Dicho código tiene esta forma: "CVE-año-código-de-cve") así como programas generados por terceras personas que aprovechen dicha vulnerabilidad. "https://github.com/" puede ser utilizado para encontrar códigos

de programación que aprovechen las vulnerabilidades obtenidas con Nuclei. De “https://github.com/” concretamente, se consiguieron los programas (Programas compuestos por un *exploit* y su respectivo *payload*) que explotan las vulnerabilidades existentes en la máquina virtual de prueba y se creó un programa para cada vulnerabilidad en python que ejecute el código en cuestión y envíe la información a un “.txt” que se pueda adjuntar al informe generado.

3.3.4. Reporte

Para la generación del informe final se utilizan los resultados obtenidos en cada una de las etapas previas. En el caso de que no haya sido posible ejecutar la fase de explotación, a partir de la fase de recolección de información también se genera el informe. Los resultados obtenidos pueden estar en formato .json, .xml y .txt. A continuación, se lista la obtención de información de los diferentes formatos.

- La información de los archivos .xml se obtiene al convertir estos archivos en tablas markdown mediante la herramienta nmap2md.
- Los datos de los archivos .json se tratan al extraer la información necesaria y almacenarla en listas.
- Los archivos .txt se tratan directamente debido a que se trata de documentos de texto plano.

Una vez obtenida la información necesaria de los archivos en diferentes formatos se crean 3 reportes en formato .pdf que corresponden a cada etapa ejecutada. En caso de que la fase de explotación no se ejecute se generan dos informes. Para generar los reportes en formato .pdf se utiliza la librería PyFPDF que genera archivos PDF utilizando python y se lee la información obtenida que se encuentra en tablas markdown, en listas y en archivos .txt. En la Figura 3.12 se presenta la clase PDF la cual crea el encabezado del archivo PDF.

```
class PDF(FPDF):
    def header(self):
        # Logo
        self.image('/src/universidad.png', 20, 8, 80) #50
        # Arial bold 15
        self.set_font('Arial', 'B', 15)
        # Move to the right
        self.cell(80)

        # Grosor del marco (1 mm)
        self.set_line_width(5)

        # Title
        #self.cell(30, 10, 'INFORME NMAP', 1, 0, 'C')
        w = self.get_string_width(title) + 6
        self.set_x((210 - w) / 2)
        #self.set_draw_color(0, 0, 0)
        self.cell(w, 60, txt = title, ln = 1, align = 'C')
```

Figura 3.12: Clase PDF para generar el encabezado del archivo PDF

Los reportes generados de cada fase se procede a unirlos usando la librería de python PdfFileMerger. El reporte final generado se envía desde gmail "pruebas1hack@gmail.com" al correo electrónico ingresado en el archivo de configuración. El módulo utilizado para el proceso de envío es el módulo smtplib. Este módulo crea un objeto de sesión de cliente *Simple Mail Transfer Protocol* (SMTP) que se utiliza para el envío de un correo electrónico a una máquina o *daemon* que escucha este protocolo. Se debe recordar que es necesario que el puerto 587 que es el de envío de mensajes de *email* esté abierto en la máquina donde se ejecuta la herramienta. Para construir el mensaje se utiliza el módulo email.mime. Debido a que el informe se envía como un archivo adjunto se crea un MIMEMultipart que funciona como un contenedor. Para enviar el PDF como un anexo se utiliza el comando "msg.attach()". Finalmente, es necesario enviar el mensaje utilizando el comando "sendmail(fromaddr, toaddr, text)" y cerrar la conexión. El análisis de los reportes se realiza en el el capítulo correspondiente a Resultados.

3.4. Conclusiones

Con el estudio de las herramientas existentes para cada fase de una prueba de penetración se determinó con cuáles se trabaja en el *framework* ya que no todas las herramientas posibilitan la automatización. Con el proceso de análisis se determinó de manera tangible que módulos se implementarán en el *framework* en base a su funcionalidad y resultados obtenidos.

Mediante la investigación acerca de las herramientas fue posible desarrollar el funcionamiento del programa principal debido a que cada herramienta produce una salida en específico y esta tiene que ser de utilidad en fases siguientes y para el reporte. El programa principal se creo para ser un gestor de módulos.

En la penúltima fase del *framework* gracias a la investigación realizada, se concluye que la mejor manera de integrar ataques que se ejecuten de forma automática en una herramienta de pruebas de penetración es mediante *scripts* de terceras personas. Este proceso permite que se elimine la interacción con el usuario.

Con la instalación del *framework* dentro de un contenedor Docker se facilita en gran medida su utilidad fuera del entorno de desarrollo. El contenedor representa una gran ventaja para la realización de pruebas, especialmente en máquinas virtuales con acceso limitado (sin interfaz gráfica). Además, permite la fácil instalación de las herramientas y librerías utilizadas. A su vez, la herramienta Cron programa la ejecución de análisis en tiempos específicos, lo que permite la comprobación continua de la seguridad informática de la red de la universidad.

Capítulo 4

Pruebas de penetración y resultados

Este capítulo presenta los resultados obtenidos de los reportes generados por el *framework*. En primer lugar, se realiza la descripción de los escenarios en donde se realizan los análisis (Sección 4.1). A continuación se presenta un análisis del reporte obtenido en un entorno controlado (Sección 4.2). Luego, se detallan los resultados del reporte obtenido en la red interna de la Universidad de Cuenca (Sección 4.3). Finalmente, se analiza el informe de la red de la Universidad de Cuenca desde CEDIA (Sección 4.4).

4.1. Descripción de los escenarios

El *framework* desarrollado analiza la red interna de la Universidad de Cuenca. Esta red pertenece a la red de universidades CEDIA que interconecta a las instituciones universitarias. La red CEDIA está constituida por un anillo de fibra óptica que tiene un ancho de banda de 100 Gbps para los establecimientos pertenecientes. Esta red es parte de la Cooperación Latino Americana de Redes Avanzadas (CLARA) que se compone de redes similares a CEDIA en Latinoamérica. A su vez, la red CLARA se interconecta con las Redes Avanzadas internacionales existentes en América del Norte, Europa, Asia y África [79].

La topología de la red de la Universidad de Cuenca se presenta en la Figura 4.1. En el presente trabajo de titulación se realizan dos pruebas utilizando el *framework* desarrollado: en primer lugar se realiza una prueba de penetración externa desde la red CEDIA, y en segundo lugar una prueba de penetración interna desde la Universidad de Cuenca. Para la prueba externa se utiliza una máquina virtual con el sistema operativo Ubuntu 20.04 mediante una conexión SSH la cual tiene acceso a la red CEDIA. Por otro lado, la prueba interna se realiza usando una máquina virtual Ubuntu 20.04 desde la red de servidores de la Universidad de Cuenca.

En primer lugar, se realizó una evaluación del *framework* utilizando un entorno virtual con una máquina virtual vulnerable. La máquina virtual Metasploitable constituye un sistema operativo y una aplicación web que posee todo tipo de vulnerabilidades de manera que cualquier usuario puede intentar explotarlas y experimentar (entrenar) [51]. Al realizar un proceso de *Hacking* ético/*Red team*

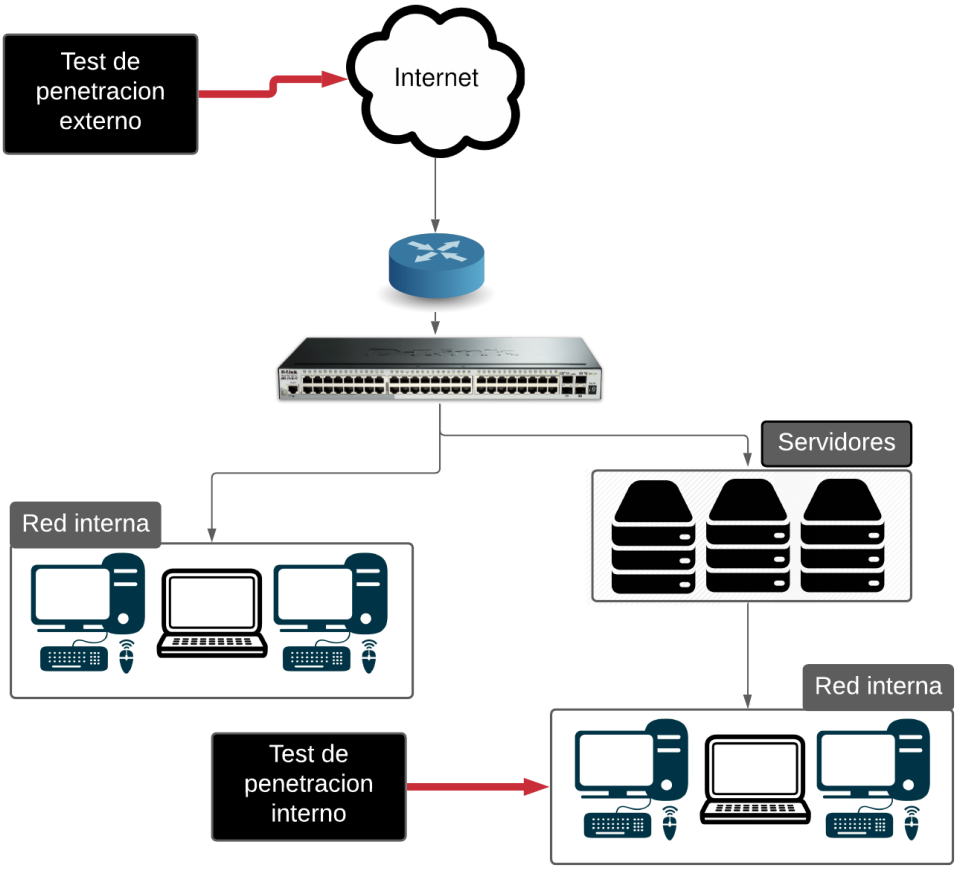


Figura 4.1: Topología de la red

no se debe realizar pruebas directamente en un entorno real. Por lo cual lo correcto es el uso de una máquina virtual vulnerable (Metasploitable) o un sistema prueba creado con base en cualquier sistema operativo con las actualizaciones desactivadas [51]. La ejecución del *framework* considerando como objetivo la máquina vulnerable demuestra la efectividad del sistema en un entorno controlado. Esta etapa se realizó previo a los análisis a realizar en la Universidad de Cuenca.

4.2. Análisis del reporte obtenido en un entorno controlado

En el análisis externo se obtuvo la dirección IP y la *Media access control* (MAC).

IP: 192.168.0.110 MAC: 08:00:27:42:51:79

También, se consiguieron los puertos y servicios asociados al objetivo (ver Tabla 4.1).

4.2.1. Vulnerabilidades encontradas

A partir de los datos encontrados, en la Figura 4.2 se realiza una interpretación de las vulnerabilidades encontradas según la severidad que representan. Del total de vulnerabilidades encontradas (3 resultados)

Tabla 4.1: Puertos y servicios encontrados

Puerto	Estado	Servicio
139/tcp	Abierto	Netbios-ssn
21/tcp	Abierto	Proftpd:1.3.5
22/tcp	Abierto	Openssh:6.6.1p1
3306/tcp	Abierto	Mysql-vuln-cve2012-2122
445/tcp	Abierto	Microsoft-ds
631/tcp	Abierto	CUPS/1.7 IPP/2.1
6667/tcp	Abierto	Irc-unrealircd-backdoor
80/tcp	Abierto	Apache/2.4.7 (Ubuntu)
8080/tcp	Abierto	Jetty(8.1.7.v20120910)

el 38 % suponen un riesgo crítico, el 37 % (3 resultados) poseen una criticidad alta. Finalmente, el 25 % (2 resultados) son de criticidad baja.

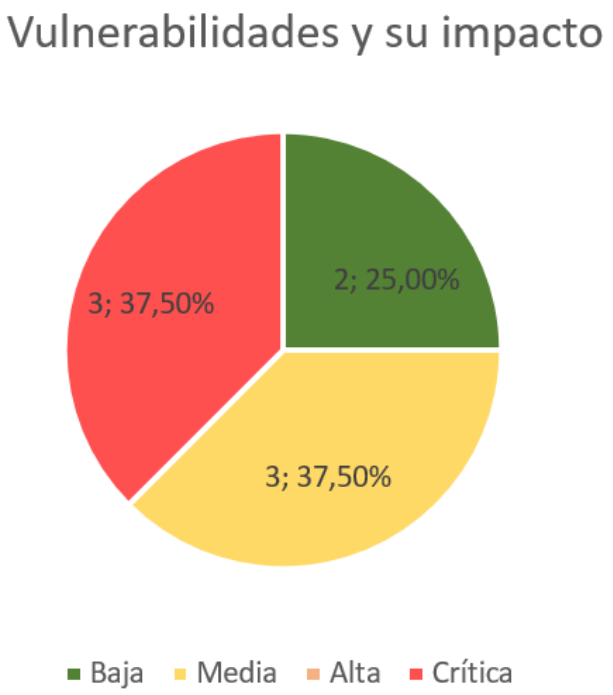


Figura 4.2: Gráfica de pastel sobre la severidad de las vulnerabilidades

Las vulnerabilidades encontradas en el *host* así como las herramientas que las obtuvieron se presentan en la Tabla 4.2.

Tabla 4.2: Vulnerabilidades encontradas.

Vulnerabilidad	Cantidad	Severidad	Herramienta
Smb-v1-Detection	1	Baja	Nuclei
CVE-2018-15473	1	Baja	Nuclei
CVE-2015-3306	1	Crítica	Nuclei
CVE-2012-2122	1	Media	NMAP
Irc-unrealircd-backdoor	1	Crítica	NMAP
CVE-2015-1158	1	Crítica	NMAP
CVE-2018-6553	1	Media	NMAP
CVE-2014-0226	1	Media	NMAP

A continuación, se presentan las vulnerabilidades reconocidas en relación con el servicio asociado, su definición y el riesgo que representan.

4.2.2. Detección SMB-V1

Es un protocolo usado principalmente para compartir archivos, servicios de impresoras y comunicación de computadoras en la red. Existen varias vulnerabilidades relacionadas a SMB. Entre las vulnerabilidades más comunes están: conseguir información detallada acerca del objetivo y conseguir acceso total al objetivo .

Recomendación: prescindir del servicio SMB-v1 y así parchear la vulnerabilidad SMB. También, se recomienda instalar constantemente las actualizaciones de seguridad [80].

4.2.3. Open SSH 6.6.1p1

La vulnerabilidad **CVE-2018-15473** existe en los servidores Open SSH antes de la versión 7.7. Estos servidores están expuestos a que agentes externos puedan extraer los nombres de usuario existentes [81].

Recomendación: Configurar el *firewall* para que limite la cantidad de conexiones que ingresan al servidor SSH. Con esta configuración se requiere una nueva conexión TCP por cada usuario probado. Esta configuración también protege contra ataques de fuerza bruta hacia las credenciales SSH [82].

4.2.4. Proftpd 1.3.5

El servicio Proftpd 1.3.5 da paso a la vulnerabilidad **CVE-2015-3306**. Esta debilidad en la seguridad informática permite a los atacantes leer y escribir en archivos arbitrarios dentro del servidor, lo que permite la implantación de *software* malicioso, de manera que se pueda obtener un Shell y control total del sistema vulnerable.

Recomendación: cambiar la versión de Proftpd a la 1.3.5a, 1.3.6rc1 o más actual [83].

4.2.5. MySQL

El Servicio MySQL esta asociado a la vulnerabilidad **CVE-2012-2122**, la base de datos MySQL con versiones 5.1.x anterior a 5.1.63, 5.5.x anterior a 5.5.24, y 5.6.x anterior a 5.6.6. Permite que

los atacantes eviten la autenticación utilizando repetidamente la misma contraseña incorrecta. Este proceso provoca una comparación con resultado positivo en una variable de entorno no validada [84]. **Recomendación:** migrar a una versión mas reciente de MySQL, superior a la 5.6.6 [85].

4.2.6. UnrealIrcd

El programa Unreal Ircd (vulnerabilidad Irc-unrealircd-backdoor) presenta una puerta trasera añadida en la versión IRCD 3.2.8.1. Esta vulnerabilidad permite acceder al sistema objetivo, teniendo control total de las funciones del usuario asociado.

Recomendación: descargar nuevamente el software UnrealIRCD y verificar su integridad utilizando su código *Secure Hash Algorithm* (SHA)1/*Message-Digest Algorithm 5* (MD5) y reinstalarlo [86].

4.2.7. CUPS/1.7 IPP/2.1

Asociado a este servicio se encuentran diversos tipos de vulnerabilidades. Entre ellas están: XSS, ejecución de código y varios tipos de *Denial of service* (DoS). Otra vulnerabilidad es la **CVE-2015-1158**, que permite a los *crakers* enviar información corrupta de cadenas de texto de referencia a través de: (1) IPP_CREATE_JOB o (2) petición IPP_PRINT_JOB y reemplazando el archivo de configuración y ejecutando código arbitrario [87].

Recomendación: deshabilitar la interfaz de web de CUPS reduce significativamente el impacto de esta vulnerabilidad [88].

También, se considera la vulnerabilidad **CVE-2018-6553**. Esta debilidad consiste en el descubrimiento de un *bypass* de caja de arena *AppArmor* en cups debido al uso de enlaces físicos que no están cubiertos por el perfil de *AppArmor*. Un atacante podría usar el enlace físico, si existe, para ejecutar los *scripts* internos sin restricciones de *sandbox*. Esta laguna en la seguridad afecta a las versiones anteriores a 2.2.7-1ubuntu2.1 en Ubuntu 18.04, antes de 2.2.4-7ubuntu3.1 en Ubuntu 17.10, antes de 2.1.3-4ubuntu0.5 en Ubuntu 16.04 y antes de 1.7.2-0ubuntu1.10 en Ubuntu 14.04 [89].

Recomendación: cambiar la versión de CUP a una superior a la 2.2.4-7ubuntu2.1 [90].

4.2.8. Apache 2.4.7

La condición de carrera del módulo mod_status del servidor de Apache antes de la versión 2.4.10 (**CVE-2014-0226**) permite que usuarios remotos maliciosos causen denegación de servicios (Desbordamiento de *buffer* basado en pila memoria) [91].

Recomendación: migrar a una versión actualizada de Apache superior a la 2.4.10 [92].

4.2.9. Explotación realizada por el *framework*

El *framework* obtiene los códigos de los CVEs posibles mediante Nuclei. El ataque se realiza utilizando los códigos de los CVEs y los programas para la explotación relacionados a las CVEs. La información del ataque realizado a la **CVE-2018-15473** acerca de los usuarios existentes en un servidor SSH se presenta en la Figura 4.3. También, se muestra parte de los resultados debido a la longitud total del resultado obtenido.

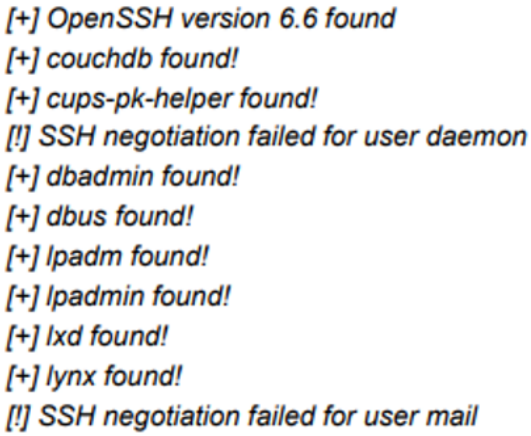


Figura 4.3: Captura del ataque realizado presentado por el *framework*

Finalmente, en el reporte generado por el *framework* se visualiza un segundo ataque realizado correspondiente a la vulnerabilidad **CVE-2015-3306**. Mediante esta vulnerabilidad se obtiene un *shell* en el sistema objetivo con el propósito de obtener control total. El *framework* presenta la creación de la *shell* asociada, la cual se encuentra en el objetivo y se puede acceder mediante una URL (véase la Figura 4.4).

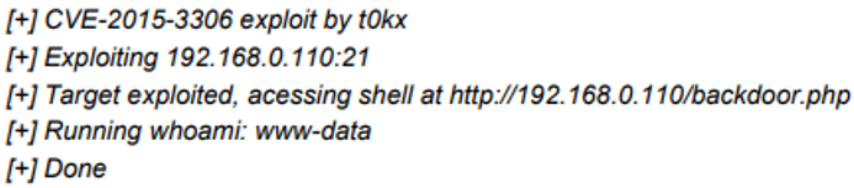


Figura 4.4: Captura del segundo ataque realizado y presentado por el *framework*

4.3. Análisis del reporte obtenido en la red interna de la Universidad de Cuenca

Al analizar el reporte generado de la prueba de penetración de red interna se encontraron algunas IPs relacionadas con subdominios de la Universidad. En los subdominios se encontraron algunos tipos de vulnerabilidades, con propósito de proteger los datos de la entidad se presentan únicamente las vulnerabilidades encontradas sin relacionarlas con la dirección IP o subdominio.

En la Figura 4.5 se presentan las vulnerabilidades obtenidas en función de la criticidad que suponen. De la totalidad de vulnerabilidades encontradas por el *framework* el 12 % (2 resultados) son de criticidad alta, el 12 % (2 resultados) representan un riesgo bajo. Finalmente, el 76 % (13 resultados) está conformado por vulnerabilidades que suponen un riesgo medio para la red interna de la Universidad de Cuenca.

Las vulnerabilidades encontradas en la red interna así como las herramientas que las obtuvieron se presentan en la Tabla 4.3. En las Secciones 4.3.1 a la 4.3.6 se detallan las vulnerabilidades reconocidas

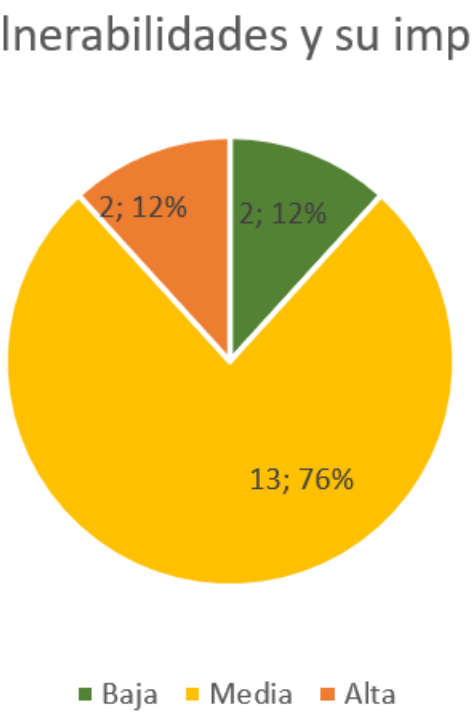


Figura 4.5: Gráfica de pastel sobre la severidad de las vulnerabilidades encontradas en la red interna

en función del servicio asociado, su definición y el riesgo que representan.

Tabla 4.3: Vulnerabilidades encontradas en la red interna

Vulnerabilidad	Cantidad	Severidad	Herramienta
CVE-2018-15473	2	Baja	Nuclei
CVE-2015-5600	1	Media	NMAP
Vulnerabilidades CSRF	2	Media	NMAP
Inyección SQL	2	Media	NMAP
CVE-2021-32785	2	Alta	NMAP
CVE-2014-0226	2	Media	NMAP
CVE-2021-32028	2	Media	NMAP
CVE-2007-6750	4	Media	NMAP

4.3.1. OpenSSH 7.4

La vulnerabilidad **CVE-2018-15473** corresponde a la versión de OpenSSH encontrada se detalló en la Sección 4.2.3.

4.3.2. Openssh 6.7p1

Con un código CVE, **CVE-2015-5600**, *Open* SSH previo a la versión 6.9 no restringe adecuadamente el procesamiento de dispositivos interactivos como teclados dentro de una sola conexión, lo que

facilita a usuarios malintencionados realizar ataques de fuerza bruta o causar una DoS.

Recomendación: deshabilitar el método de autenticación interactiva que se puede lograr agregando la linea "ChallengeResponseAuthentication no" al archivo "/etc/ssh/sshd_config" y reiniciando el servicio SSH [93].

4.3.3. Vulnerabilidades CSRF

Los riesgos de la explotación de esta vulnerabilidad incluyen distintos tipos de actividades ilícitas e indeseadas. Desde el acceso a cuentas privadas de usuarios, así como acusar a alguien de acceder a sitios de contenido delicado hasta habilitar algún tipo de filtro o regla en el correo electrónico para que todos los correos se reenvíen a otra cuenta [94].

Recomendación: utilizar un *token* secreto. Este proceso consiste en la inclusión de un valor aleatorio que se genera y se informa al navegador del usuario en el momento que se inicie sesión. También, se puede utilizar el envío doble de *Cookies* que es una variante al *token* donde el código corresponde a la *cookie* de sesión de usuario. El servidor debe comprobar que las dos sean iguales en cada solicitud. Por último, se puede utilizar un *captcha* o limitar el tiempo de vida de las sesiones [95].

4.3.4. Inyección SQL

Esta vulnerabilidad es explotable a través de la inyección de códigos SQL en instrucciones SQL durante el ingreso de datos a través de una página web. Durante la ejecución de una instrucción SQL, una carga maliciosa controla el servidor de base de datos de la aplicación web. La severidad de esta vulnerabilidad puede ir desde baja hasta crítica. Se determina como severidad media debido a que no se realizó la explotación de la vulnerabilidad.

Recomendación: utilizar sentencias preparadas, lo que significa que en lugar de ensamblar directamente la cadena de consulta y ejecutarla, se almacena una sentencia preparada, se alimenta con los datos y la ensambla y sanitiza para su ejecución [96].

4.3.5. Apache http server 2.4.6

A más de la vulnerabilidad **CVE-2014-0226** descrita en la Sección 4.2.8, existen otras vulnerabilidades que afectan a un servidor Apache de menor versión a la 2.4.10. Esta vulnerabilidad actúa sobre la condición de carrera del módulo mod_status que permite que atacantes remotos causen DoS basándose en el desbordamiento de pila de memoria o posiblemente para tener acceso a información, credenciales o ejecutar código arbitrario.

Recomendación: migrar a una versión de apache superior o igual a la 2.4.10 [97].

Otra vulnerabilidad derivada al servicio Apache 2.4.6 es la **CVE-2021-32785** que se aplica en versiones anteriores a la 2.4.9. El módulo mod_auth_openidc está configurado para utilizar caché no encriptado, la interpolación de argumentos realizada de forma errónea antes de pasar las peticiones ocasiona que la acción se vuelva a realizar y conduce a un error de cadena de formato descontrolado. Este proceso puede provocar una DoS.

Recomendación: actualizar a la versión 2.4.9 y realizar la interpolación de argumentos utilizando la API hiredis [98].

4.3.6. Postgresql 9.6

Este servicio tiene la vulnerabilidad **CVE-2021-32028**. Mediante esta vulnerabilidad un atacante puede leer bytes arbitrarios de la memoria del servidor. En la configuración por defecto, cualquier usuario de base de datos autenticado puede crear objetos de prerrequisitos y completar este ataque a voluntad. Un usuario que carece de los privilegios *CREATE* y *TEMPORARY* en todas las bases de datos y el privilegio *CREATE* en todos los esquemas no podrá utilizar este ataque a voluntad. Cabe recalcar que no es la única vulnerabilidad conocida de este servicio.

Recomendación: actualizar el sistema de gestión de bases de datos Postgresql a las versiones 9.6.22, 10.17, 11.12, 12.7 o 13.3 [99].

4.3.7. HTTP

El ataque DoS Slowloris (**CVE-2007-6750**) intenta mantener abiertas muchas conexiones al servidor web de destino y por el mayor tiempo posible. Este proceso se consigue enviando una petición parcial. Al hacer esta petición los recursos del servidor HTTP causan una denegación de servicio.

Recomendación: actualizar a la versión más reciente de Apache HTTP *server* [100].

4.4. Análisis del reporte obtenido de la red de la Universidad de Cuenca desde CEDIA

Finalmente, se realizó un análisis desde la infraestructura de CEDIA hacia la Universidad de Cuenca en búsqueda de vulnerabilidades que se puedan encontrar desde fuera de la universidad. En los resultados se encontraron varios subdominios relacionados con el objetivo. También, se obtuvieron algunas vulnerabilidades. Con el objetivo de proteger la información de la universidad, no se presenta información referente a direcciones IP, puertos o subdominios.

En la Figura 4.6 se hace una representación de las vulnerabilidades obtenidas en función de la criticidad que suponen. De la totalidad de vulnerabilidades encontradas (7 resultados), el 14 % (1 resultado) posee una criticidad alta mientras el 86 % (6 resultados) restante representa un riesgo medio.

A partir del análisis se encontraron algunas vulnerabilidades que se presentan en la Tabla 4.4 así como las herramientas que las obtuvieron. En la tabla se referencian las vulnerabilidades que se presentaron en las secciones anteriores.

Tabla 4.4: Vulnerabilidades encontradas en la universidad desde CEDIA

Vulnerabilidad	Cantidad	Severidad	Herramienta
Vulnerabilidades CSRF	1	Media	NMAP
Inyección SQL	1	Media	NMAP
CVE-2007-6750	1	Media	NMAP
CVE-2021-32785	1	Alta	NMAP
CVE-2014-0226	1	Media	NMAP
CVE-2005-3299	1	Media	NMAP
ASP.NET DEBUG	1	Media	NMAP

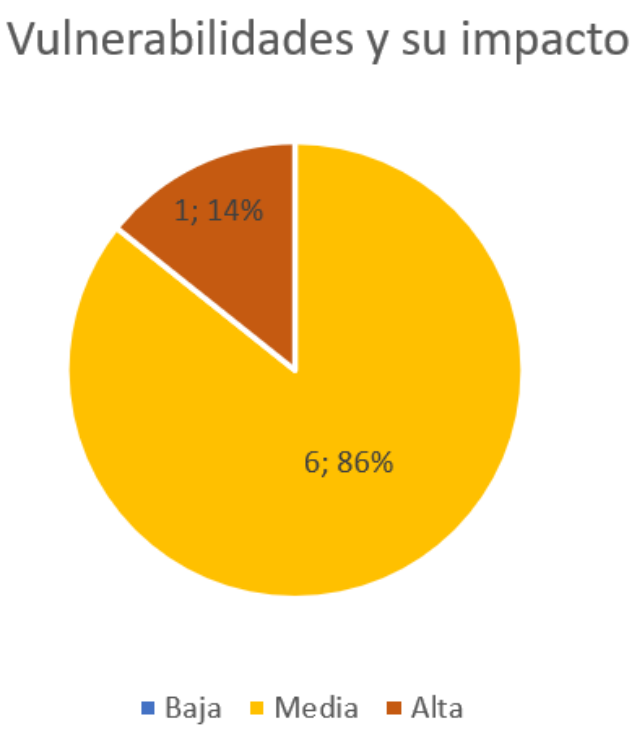


Figura 4.6: Gráfica sobre la severidad de las vulnerabilidades encontradas desde la red de CEDIA en relación a su totalidad

Cada vulnerabilidad se da por un servicio o configuración específica, por lo que se presentan los servicios y las vulnerabilidades que les corresponden. Las vulnerabilidades *Cross-site Request Forgery* (CSRF), de inyección SQL, CVE-2007-6750, CVE-2021-32785, CVE-2014-0226 se describieron en las subsecciones 4.3.3, 4.3.4, 4.3.7, 4.3.5 y 4.2.3 respectivamente.

4.4.1. Php my admin 2.6.4 pl1

La vulnerabilidad **CVE-2005-3299** de inclusión de archivos *Hypertext Preprocessor* (PHP) en grab_globals.lib.php en phpMyAdmin 2.6.4 y 2.6.4-pl1 permite a los atacantes remotos incluir archivos locales a través del parámetro *redirect*, posiblemente involucrando la matriz subformulario [84].

Recomendación: actualizar el paquete *phpmyadmin* [101].

4.4.2. ASP.NET

Con la vulnerabilidad **ASP.NET DEBUG Method Enabled** es posible enviar instrucciones de depuración a los *scripts Active Server Pages* (ASP) remotos a través del método HTTP *DEBUG*. Un atacante remoto no autenticado puede aprovechar esto para alterar el tiempo de ejecución de los *scripts* [102].

Recomendación: deshabilitar el *debug* en los archivos Web.config y Machine.config al cambiar la línea *compilation debug* a un valor *false* [103].

$$< compilationdebug = "false"/ > \tag{4.1}$$

4.5. Conclusiones

En este trabajo se ha ejecutado un análisis a la máquina virtual para comprobar el funcionamiento de la herramienta debido a que la máquina virtual Metasploitable posee vulnerabilidades conocidas previamente. Los resultados encontrados por el *framework* corresponden a las vulnerabilidades preexistentes. La herramienta consiguió explotar ciertas aperturas en la seguridad y presentar sus resultados para su análisis.

Este trabajo de titulación ha mostrado que existen algunas vulnerabilidades en la red de la Universidad de Cuenca. Los informes de los análisis realizados se entregaron a las autoridades competentes en la Universidad de Cuenca (véase Anexo B). Con base en el examen realizado desde su red interna se determinan las vulnerabilidades existentes y su severidad. Se encontraron 2 vulnerabilidades de criticidad baja, 13 de severidad media y 2 de severidad alta. Lo que da como resultado que la universidad no está expuesta a riesgos graves e inmediatos de seguridad (No se encontraron vulnerabilidades de riesgo crítico). No obstante, se han de tomar en cuenta las vulnerabilidades encontradas debido a que los hallazgos realizados pueden provocar inestabilidad o pérdida de los servicios web ofrecidos por la universidad.

El segundo hallazgo más importante fue la existencia de vulnerabilidades diferentes a las encontradas previamente mediante un análisis realizado desde la red de CEDIA. La presencia de estas vulnerabilidades se debe a los métodos utilizados para que los servicios se presenten a usuarios fuera de la red de la Universidad. Al igual que el análisis realizado a la red interna no se han encontrado vulnerabilidades críticas; sin embargo, se deben tomar acciones sobre los servicios afectados por las vulnerabilidades con el propósito de mejorar la seguridad informática de la red.

Capítulo 5

Conclusiones y trabajos futuros

Este capítulo, en afinidad con los objetivos formulados originalmente, presenta las principales conclusiones obtenidas de este trabajo de titulación (Sección 5.1). Para finalizar, se detallan algunas recomendaciones para la mejora futura del *framework* implementado (Sección 5.2).

5.1. Conclusiones

En este trabajo de titulación se ha explicado la importancia del *Hacking* ético realizado en las IES para la búsqueda de posibles vacíos existentes en la seguridad informática. Este proceso se realiza con el propósito de aminorar las vulnerabilidades existentes en el sistema y mejorar así los servicios que proveen estas instituciones. El *Hacking* ético puede ser realizado por un *Red team*, siguiendo las fases de la cadena de eliminación que propone los lineamientos para la explotación de vulnerabilidades informáticas. El *Red team* utiliza herramientas específicas que le permiten conocer al objetivo y posteriormente recolectar información necesaria para el análisis y explotación de vulnerabilidades. Mediante la revisión del estado del arte se determinó las diferentes herramientas que se utilizan comúnmente para la realización de pruebas de penetración. Todas las herramientas encontradas tienen una función específica dentro del proceso, por tal razón fueron clasificadas mediante la funcionalidad que poseen en referencia a las fases que un *Red team* utiliza.

Para el proyecto se han investigado las herramientas que se pueden incluir en el *framework*. Esta investigación se basa en la premisa que la herramienta desarrollada se debe ejecutar de forma automática debido a que se realiza un análisis periódico del estado de la red interna de la universidad. La primera fase que se ejecuta corresponde a la obtención de información y se divide en tres etapas: recopilación pasiva, activa y la identificación de vulnerabilidades. En la recopilación pasiva de información se obtiene información acerca de la institución utilizando las herramientas The Harvester y mediante NMAP se ejecuta un escaneo pasivo de los *hosts*. Para la recopilación activa igualmente se utiliza NMAP para descubrir el estado de los puertos. Finalmente, en la identificación de vulnerabilidades se utiliza Nuclei. Las fases de explotación y reporte están diseñadas mediante el uso de código en Python de terceros y codificación propia. En consecuencia, las herramientas seleccionadas para integrar el *framework* son:

The Harvester, NMAP y Nuclei debido a que permiten la ejecución de las fases sin necesidad de la interacción con el usuario. La selección de las herramientas a utilizar descartó software como Shodan, Wireshark, Nessus, OpenVas debido a que no permitían la automatización del *framework*.

En este proyecto se evaluó el sistema mediante la utilización de un entorno de pruebas, compuesto por una máquina virtual vulnerable y una máquina virtual atacante con el *framework* implementado. Bajo este entorno virtual, se comprobó el funcionamiento de la herramienta desarrollada, debido a que se obtuvo vulnerabilidades que se conocía de su existencia con anterioridad. La herramienta fue desarrollada en un contenedor Docker lo que facilita su instalación en diversidad de sistemas y situaciones, ya que no requiere interfaz gráfica para funcionar. El *framework* desarrollado presenta un reporte acerca de las vulnerabilidades existentes en la red interna de la Universidad de Cuenca de acuerdo con los lineamientos de la metodología de pruebas de penetración PTES.

La herramienta fue implementada en un servidor virtual proporcionado por las autoridades competentes sin interfaz gráfica, desde el cual se realizaron las pruebas de penetración tanto interna como externa de la red de la Universidad de Cuenca. Luego del análisis del reporte generado por la herramienta al utilizarla en la Universidad de Cuenca desde la red interna se determina la seguridad informática en el escenario en que un atacante tenga acceso a las instalaciones. Para cubrir otras posibilidades se realizó un análisis desde la red externa. Dicho análisis se ejecutó desde la infraestructura de CEDIA lo que simula un atacante externo. Similar al análisis realizado desde la red interna, se usó una máquina virtual configurada y con la herramienta funcionando. Mediante el examen externo a la red se determinan los posibles escenarios que puede utilizar un *cracker* para atacar los activos de la universidad.

El reporte final generado está conformado por los resultados de cada etapa ejecutada. Debido a que cada herramienta tiene un formato diferente salida: The Harvester, Nuclei presentan sus resultados en formato JSON y NMAP emplea el formato XML la información se extrae de formas diferentes. Los datos de los archivos JSON se obtienen utilizando listas, mientras que los archivos con extensión `.xml` se pasan a tablas markdown para procesarlos. En el reporte final se organiza la información obtenida de las herramientas y se presenta en formato PDF. Por tanto, el *framework* compila los resultados de cada herramienta y genera un informe final que se envía al *pentester*. Este estudio ha mostrado que recolectar la información de las herramientas del *framework* puede ser un proceso complicado debido a los diferentes formatos de salida de los resultados. Además, es necesario el envío periódico de reportes para conocer de forma periódica el estado de la red.

Los resultados de esta investigación muestran que la seguridad de la red interna de la universidad se debe considerar como un aspecto fundamental. Mediante el análisis realizado utilizando el *framework* Daniz, se determinó que la Universidad de Cuenca se encuentra expuesta a un nivel medio de riesgo a los ciberataques, debido a que no se encontraron vulnerabilidades críticas y solo un tipo de vulnerabilidad con criticidad alta. No obstante, se recomienda tomar acciones de seguridad con el fin de remediar las vulnerabilidades encontradas.

5.2. Trabajos Futuros

Se recomienda que estudios futuros sobre el presente tema consideren agregar nuevas funcionalidades, al aumentar el número de herramientas existentes en cada una de las fases y subfases que componen el *framework* con el objetivo de descubrir una mayor cantidad de información y vulnerabilidades

existentes en la red interna de la universidad. También, se propone actualizar la herramienta de forma constante a través de la integración de nuevos ataques prediseñados con el propósito de explotar las vulnerabilidades asociadas.

Sería conveniente realizar un mayor trabajo en la generación del reporte final. En este aspecto se sugiere la incorporación de un sistema de alerta en el caso de que un informe sugiera la existencia de vulnerabilidades de nivel crítico. Finalmente, se recomienda que estudios futuros aborden un análisis a la red administrativa de la universidad con el fin de determinar la existencia de otras vulnerabilidades.

Anexo A

Manual de usuario

En este anexo se presenta el manual de usuario del *framework* así como las dependencias necesarias para su instalación.

A.1. Instalación de dependencias necesarias

El *framework* se desarrolló en un contenedor Docker y usa los paquetes `docker`, `docker-compose` y `git`. Los comando necesarios para instalar estos paquetes se encuentran en el Listado A.1.

```
# sudo apt-get install docker
# sudo apt-get install docker-compose
# sudo apt-get install git
```

Listado A.1: Instalación de dependencias

A.2. Configuración previa a la ejecución de programa

Previo al proceso de configuración se tiene que clonar el repositorio donde se encuentra le programa (Véase listado A.2).

```
# git clone https://github.com/fabianastudillo/daniz-red.git
# cd daniz-red
```

Listado A.2: Instalación de dependencias

La herramienta requiere para su ejecución modificar el archivo de configuración que se encuentra en la dirección /kali/src, este archivo se compone de 3 líneas de información: un email válido (gmail), una interfaz de red conectada a Internet (De la maquina atacante) y el objetivo o grupo de objetivos separados por comas (véase la Figura A.1).

Además del archivo de configuración se debe modificar el archivo utilizado por Cron (crontab) que se encuentra en la dirección `kali/src`. El archivo crontab se utiliza para que el *framework* se ejecute de forma periódica. La Figura A.2 muestra un ejemplo de configuración.

Figura A.1: Archivo de configuración

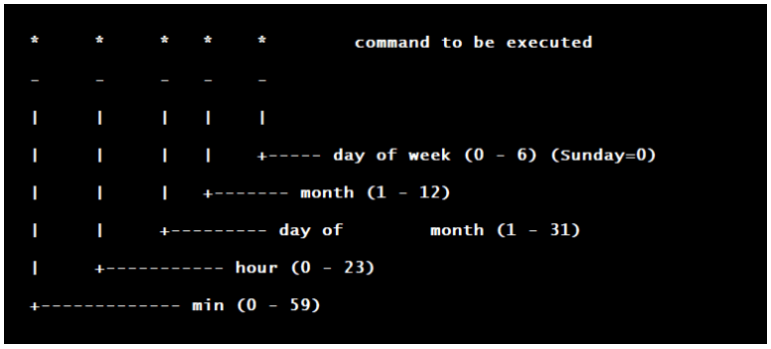


Figura A.2: Archivo crontab [4]

A.3. Ejecución del *framework*

Para ejecutar el programa, primero se tiene que clonar el repositorio de Github, luego crear la imagen Docker, y finalmente, ejecutar el programa en segundo plano o manualmente con la utilización de los comandos mostrados en el Listado A.3. Para una explicación mas ilustrativa dirigirse al enlace: https://youtu.be/m6tgQQw1ry8

```
# docker-compose build
# docker-compose up -d kali (Para q se ejecute de forma automática)
# docker exec kali /usr/bin/python3 /src/daniz-auto.py -c /src/config.txt  (Para
    ejecutarlo manualmente)
```

Listado A.3: Comando para crear la imagen Docker

Anexo B

Recepción de documentos

En este anexo se presenta la recepción por parte de la autoridad competente de los análisis realizados a la red de la Universidad de Cuenca con el *framework* Daniz 

Cuenca, 13 de septiembre de 2021

Srta. y Sr.,
Cinthya Campoverde, Brian Quinde
Egresados de la Carrera de Electrónica y Telecomunicaciones

Asunto: Recepción de los análisis de los reportes

De mi consideración:

Por medio de la presente comunico la recepción de los análisis de los reportes generados por el *framework* Daniz desarrollado por la Srta. Cinthya Lisseth Campoverde Andrade con CI: 0106510415 y el Sr. Brian Daniel Quinde Saltos con CI: 0302563259 y me encuentro conforme con los resultados.

Atentamente,

Firmado electrónicamente por:
MARIA JOSE TORRES MALDONADO

Ing. María José Torres
COORDINADORA DE REDES Y COMUNICACIONES

Bibliografía

[1] K. B. Chowdappa, S. S. Lakshmi, y P. N. V. S. P. Kumar, "Ethical hacking techniques with penetration testing," pp. 3389–3393, 214. [En línea]. Disponible: www.ijcsit.com

[2] Department of Defense Cyber Red Team Certification and Accreditation,, "About gvm architecture." [En línea]. Disponible: https://community.greenbone.net/t/aboutgvm-architecture/1231

[3] J. Martinez, "Kill Chain y Cyber Kill Chain. Defensa cibernética desde un enfoque militar," p. 3, 2019. [En línea]. Disponible: https://grupocsf.com/tag/cyber-kill-chain/

[4] T. T. Magazine, "Crontab." [En línea]. Disponible: https://www.adminschoice.com/crontab-quick-reference

[5] A. Romero, "Las redes de información y su importancia para la investigación científica," *Sedici.Unlp.Edu.Ar*, vol. 7, p. 19, 2011.

[6] F. J. Merchan, Lima, F. Astudillo-Salinas, L. Tello-Oquendo, F. Sanchez, G. Lopez y D. Quiroz, "Information security management frameworks and strategies in higher education institutions: a systematic review," vol. 7, p. 3, 2020.

[7] G. Preciado Vidal-Aragón, "El Big Data Y La Huella Digital: La Importancia De Los Datos Y Cómo Son Utilizados Por Las Empresas," vol. 7, p. 3, 2019. [En línea]. Disponible: https://repositorio.comillas.edu/xmlui/handle/11531/33063.

[8] R. Sinopsis, "Filtración en Marriott expone los datos de 500 millones de clientes | Sinopsis," 2018. [En línea]. Disponible: http://sinopsis.mx/2018/11/filtracion-en-marriott-expone-los-datos-de-500-millones-de-clientes/

[9] Metro Ecuador, "Ecuador es el país de Latinoamérica con más infecciones de 'ransomware' | Metro Ecuador," 2018. [En línea]. Disponible: https://www.metroecuador.com.ec/ec/tecnologia/2018/08/07/ecuador-pais-latinoamerica-mas-infecciones-ransomware.html

[10] J. Maldonado, "GTR 2021 de NTT: crecen hasta un 300 oportunistas dirigidos a sectores específicos - Notas de prensa," p. 1, 2021. [En línea]. Disponible: https://www.comunicae.es/nota/gtr-2021-de-ntt-crecen-hasta-un-300-los-1224823/

[11] N. Dávalos, "La ciberseguridad en el país ha mejorado, pero aún no es suficiente," *Primicias, Quito, jul 06*, p. 1, 2020.

[12] ESET, "Eset Security Report Latinoamérica 2018," pp. 3–15, 2017. [En línea]. Disponible: https://www.welivesecurity.com/wp-content/uploads/2018/06/ESET_security_report_LATAM2018.pdf

[13] Expreso, "Ecuador está en el grupo de los rezagados en ciberseguridad," 2020. [En línea]. Disponible: https://cybermap.kaspersky.com/es/stats#country=35&type=vul&period=

[14] IT ahora, "Estado Actual de la Ciberseguridad 2020," pp. 1–18, 2020. [En línea]. Disponible: https://www.itahora.com/wp-content/uploads/2020/06/ESTADO-ACTUAL-DE-CIBERSIGURIDAD-ECUADOR-2020-1.pdf.

[15] Kaspersky Lab, "Ciberamenaza, Mapa Tiempo Real," 2017. [En línea]. Disponible: https://cybermap.kaspersky.com/es/stats#country=35&type=vul&period=

[16] Julio Navarrete, "ECUADOR EN RIESGO CIBERATAQUES BDO," 2020. [En línea]. Disponible: https://www.bdo.ec/es-ec/noticias/2020/ecuador-en-riesgo-ciberataques

[17] M. Sistemas, "Políticas de seguridad de la información: consecuencias de su falta - blog mxm," 7 2016. [En línea]. Disponible: https://www.mxm.com.br/es/blog/politicas-de-seguranca-da-informacao-consequencias-da-sua-falta/

[18] O. M. Oñate Haro and Martinez Cáceres, "Mejoras en la seguridad de la red inalámbrica de la Universidad Nacional de Chimborazo aplicando hacking ético," *Universidad Nacional de Chimborazo,*, p. 1, 2017.

[19] G. P. Vidal-Aragón, "El big data y la huella digital : la importancia de los datos y cómo son utilizados por las empresas," 2019. [En línea]. Disponible: https://repositorio.comillas.edu/xmlui/handle/11531/33063

[20] M. D. Murillo, "Informe de trabajo de titulación," pp. 1–74, 2019. [En línea]. Disponible: https://www.bdo.ec/es-ec/noticias/2020/ecuador-en-riesgo-ciberataques

[21] Cedia, "Informe General de actividades CEDIA 2019," pp. 40–42, 2019. [En línea]. Disponible: https://www.cedia.edu.ec/dmdocuments/publicaciones/Informesanuales/Informeanual2019CEDIA.pdf.

[22] A. Vanegas Romero, "Pentesting, ¿Por qué es importante para las empresas?" *Univ. Pilot. Colomb.*, pp. 1–5, 2019.

[23] Karina Astudillo, "Plataformas de Explotación (Hacking Frameworks): ¿cuáles son las mejores?" [En línea]. Disponible: https://karinaastudillo.com/plataformas-de-explotacion-hacking-frameworks-cuales-son-las-mejores/

[24] NormaIso, "Evaluación del Desempeño en ISO 27001- Requisitos en detalle." [En línea]. Disponible: https://normaiso27001.es/evaluacion-del-desempeno-en-iso-27001/

[25] D. Arcentales Fernández y X. Caycedo Casas, "Auditoría informática: un enfoque efectivo," *Dominio de las Ciencias*, vol. 3, num. 3 mon, pp. 157–173, 2017.

[26] M. E. Alzahrani, "Auditing Albaha University Network Security using in-house Developed Penetration Tool," *Journal of Physics: Conference Series*, vol. 978, num. 1, 2018.

[27] E. Chow, "Ethical Hacking & Penetration Testing," pp. 1–37, 2011. [En línea]. Disponible: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.475.3877&rep=rep1&type=pdf.

[28] Joseph Plot, Alan Shaffer, Gurminder Singh, "Cyber Automated Red Team Tool," *Proceedings of the 53rd Hawaii International Conference on System Sciences*, vol. 1, p. 2, 2020.

[29] P. Mauricio Brito Bermúdez, "Diagnóstico de ethical hacking para la universidad politécnica salesiana," p. 122.

[30] E. Parra Tapia, "Identificación y análisis de vulnerabilidades en los portales web de la universidad politécnica salesiana a través de técnicas de pentesting," pp. 1–10.

[31] C. Wong y S. Mike, "Pen test metrics 2018," 2 2018. [En línea]. Disponible: https://resource.cobalt.io/hubfs/Pen%20Test%20Metrics%202018.pdf

[32] R. Moore, "Cybercrime: Investigating high-technology computer crime - 2nd edition," 10 2010. [En línea]. Disponible: https://www.routledge.com/Cybercrime-Investigating-High-Technology-Computer-Crime/Moore/p/book/9781437755824#

[33] T. Wilhelm, "Professional penetration testing: Creating and learning in a hacking lab," pp. 0–0, 2013.

[34] R. A. Española, "jáquer | Definición | Diccionario de la lengua española | RAE - ASALE," 2020. [En línea]. Disponible: https://dle.rae.es/j{á}quer#TLIznqw

[35] Guevara Soriano A, "Hacking ético: mitos y realidades. Revista .Seguridad. 2012," 2012. [En línea]. Disponible: https://revista.seguridad.unam.mx/numero-12/hacking-%C3%A9tico-mitos-y-realidades

[36] C. C. Palmer, "Ethical hacking," *IBM Systems Journal*, vol. 40, pp. 769–780, 3 2001. [En línea]. Disponible: https://dl.acm.org/doi/abs/10.1147/sj.403.0769

[37] E. F. Medina Rojas, "Hacking Ético: Una herramienta para la seguridad informática," *Universidad Piloto de Colombia*, pp. 1–8, 2015.

[38] A. Rojas, R. Bautista, y C. Medina, "Pentesting empleando técnicas de ethical hacking en redes IPv6," *Revista Ingenio UFPSO*, vol. 11, pp. 79–96, 2016.

[39] Luis Alcides Mendaño, "Implementaciòn de tècnicas de hacking etico para el descubrimiento y evaluaciòn de vulnerabilidades de la red de una cartera de estado," Ph.D. dissertation, Universidad Politecnica Nacional, 2016.

[40] EC-Council, "Ec-council ec-council ethical hacking and countermeasures." [En línea]. Disponible: http://www.eccouncil.orghttp://www.eccouncil.org

[41] E. O. Diogenes Yuri, *Cybersecurity: Attack and Defense Strategies*. Birmingham-Mumbai: Packt Publishing, 2018.

[42] U. Revista, "Red team, blue team y purple team: funciones y diferencias," 1 2020. [En línea]. Disponible: https://www.unir.net/ingenieria/revista/red-blue-purple-team-ciberseguridad/

[43] Department of Defense Cyber Red Team Certification and Accreditation,, "Cjcsm 6510.03, department of defense, washington, dc, usa, 2013." [En línea]. Disponible: https://www.jcs.mil/Portals/36/Documents/Library/Manuals/m651003.pdf?ver=2016-02-05-175711-083.

[44] E. Hutchins, M. Cloppert, y R. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *6th International Conference on Information Warfare and Security, ICIW 2011*, num. July 2005, pp. 113–125, 2011.

[45] Tenable, "Evaluación de vulnerabilidades Nessus | Tenable®." [En línea]. Disponible: https://es-la.tenable.com/products/nessus

[46] B. P. Angela Orebaugh, *Nmap In the Enterprise Your Guide to Network Scanning*, 1ra ed. syngress, 208.

[47] "laramies/theHarvester: E-mails, subdomains and names Harvester - OSINT." [En línea]. Disponible: https://github.com/laramies/theHarvester

[48] J. D. Fernández-Núñez, "Análisis de vulnerabilidades de una red corporativa mediante Shodan y OpenVAS," Ph.D. dissertation, Centro Universitario de la Defensa en la Escuela Naval Militar.

[49] "projectdiscovery/nuclei: Fast and customizable vulnerability scanner based on simple YAML based DSL." [En línea]. Disponible: https://github.com/projectdiscovery/nuclei

[50] Ruben Velasco, "OWASP ZAP, herramienta para auditar la seguridad de una página web," p. 1, 2015. [En línea]. Disponible: https://www.redeszone.net/2015/04/25/seguridad-web-owasp-zap/

[51] A. E. Rodríguez Llerena, "Herramientas fundamentales para el hacking ético," *Revista Cubana de Informática Médica*, vol. 12, num. 1, pp. 116–131, 2020.

[52] F. Pastor Ricós, "Pentesting y generación de exploits con Metasploit," Ph.D. dissertation, Universitat Oberta de Catalunya, 2020.

[53] D. Rountree, "Privilege Escalation - an overview | ScienceDirect Topics," 2011. [En línea]. Disponible: https://www.sciencedirect.com/topics/computer-science/privilege-escalation

[54] "Auditorías y análisis de redes e infraestructuras | Ciberseguridad." [En línea]. Disponible: https://ciberseguridad.com/servicios/auditorias-analisis-redes-infraestructuras/

[55] Lockheed Martin, "Gaining the Advantage - Applying Cyber Kill Chain Methodology to Network Defense," pp. 1–13, 2015. [En línea]. Disponible: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf

[56] E. Hutchins, M. Cloppert, y R. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *6th International Conference on Information Warfare and Security, ICIW 2011*, num. July 2005, pp. 113–125, 2011.

[57] Cisco Network Academy, *Introducción a la Ciberseguridad*. [En línea]. Disponible: https://static-course-assets.s3.amazonaws.com/CyberSec2.1/es/index.html{#}4.2.2.1

[58] E. Chow, "Ethical Hacking & Penetration Testing," *IT Research Paper*, pp. 1–37, 2011. [En línea]. Disponible: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.475.3877&rep=rep1&type=pdf

[59] P. Engebretson, *The basics of Hacking and Penetration testing*, H. Scherer, Ed., Waltham, 2011.

[60] K. V. Alvarez Intriago y A. C. Gamboa, "Propuesta de una Metodología de Pruebas de penetración orientada a riesgos," num. c, pp. 1–25, 2018.

[61] K. Basuki, "Diseño de un marco de referencia para el análisis de vulnerabilidades a un segmento de la red corporativa de una empresa de telecomunicaciones en Quito basado en las principales metodologias de pruebas de seguridad informática," *ISSN 2502-3632 (Online) ISSN 2356-0304 (Paper) Jurnal Online Internasional & Nasional Vol. 7 No.1, Januari – Juni 2019 Universitas 17 Agustus 1945 Jakarta*, vol. 53, num. 9, pp. 1689–1699, 2019. [En línea]. Disponible: www.journal.uta45jakarta.ac.id

[62] Elkan, M., "Auditoría de seguridad, hacking ético y prueba de penetración. ," 2017. [En línea]. Disponible: http://blogdeauditoriadeseguridad.blogspot.com/2017/09/metodologia-ptes-penetration-testing.html

[63] Veracode, "Owasp top 10 vulnerabilities | veracode." [En línea]. Disponible: https://www.veracode.com/security/owasp-top-10

[64] A. Chowdhury, "Recent cyber security attacks and their mitigation approaches – an overview," in *Applications and Techniques in Information Security*, L. Batten y G. Li, Eds. Singapore: Springer Singapore, 2016, pp. 54–65.

[65] J. Veloz, A. Alcivar, G. Salvatierra, y C. Silva, "Ethical hacking, una metodología para descubrir fallas de seguridad en sistemas informáticos mediante la herramienta KALI-LINUX," *Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones*, vol. 1, num. 1, pp. 1–12, 2017.

[66] J. F. Gimeno, "Análisis, diseño y desarrollo de una aplicación para la realización automática de pentesting," Ph.D. dissertation, Escuela Politecnica Superior, 2019.

[67] Juliana Zapata Garcia, "USO DE TECNOLOGÍAS DE PRUEBAS DE PENETRACIÓN PARA VALIDACIÓN DE SEGURIDAD DE APLICACIONES WEB BASADO EN EL TOP 10 DE VULNERABILIDADES DE OWASP," *Energies*, vol. 6, num. 1, pp. 30–41, 2018. [En línea]. Disponible: http://journals.sagepub.com/doi/10.1177/1120700020921110{%}0Ahttps://doi.org/10.1016/j.reuma.2018.06.001{%}0Ahttps://doi.org/10.1016/j.arth.2018.03.044{%}0Ahttps://reader.elsevier.com/reader/sd/pii/S1063458420300078?token=C039B8B13922A2079230DC9AF11A333E295FCD8

[68] CHICAIZA RAMÍREZ SABRINA ESTEFANÍA, "Universidad Politécnica Salesiana Sede Quito," *Tesis*, pp. 1–30, 2018. [En línea]. Disponible: http://dspace.ups.edu.ec/bitstream/123456789/5081/1/UPS-CYT00109.pdf

[69] Herzog, P., "Manual de la metodología abierta de testeo de seguridad OSSTMM," 2003. [En línea]. Disponible: http://fcbi.unillanos.edu.co/segurinfo.unillanos/archivos/materialApoyo/OSSTMM.es.2.1.pdf

[70] D. Tenable, "Generate an API Key (Nessus)." [En línea]. Disponible: https://docs.tenable.com/nessus/Content/GenerateAnAPIKey.htm

[71] AWS, "Contenedores de docker." [En línea]. Disponible: https://aws.amazon.com/es/docker/

[72] D. docs, "Docker overview | docker documentation." [En línea]. Disponible: https://docs.docker.com/get-started/overview/

[73] D. S. Orcero, *Kali Linux*. Madrid: Ra-Ma, 2018.

[74] K. L. Documentation, "What is kali linux?" [En línea]. Disponible: https://www.kali.org/docs/introduction/what-is-kali-linux/

[75] P. S. Foundation, "What is python?" [En línea]. Disponible: https://www.python.org/doc/essays/blurb/

[76] Rapid7, "Free Metasploitable Download: Evaluate Metasploit with Our Target Machine." [En línea]. Disponible: https://information.rapid7.com/metasploit-framework.html

[77] C. Wei, "Metasploitable3: An intentionally vulnerable machine for exploit testing," 11 2016. [En línea]. Disponible: https://www.rapid7.com/blog/post/2016/11/15/test-your-might-with-the-shiny-new-metasploitable3/

[78] Eset, "Auditando con Nmap y sus scripts para escanear vulnerabilidades | WeLiveSecurity." [En línea]. Disponible: https://www.welivesecurity.com/la-es/2015/02/12/auditando-nmap-scripts-escanear-vulnerabilidades/

[79] S. de la Red CEDIA, "Servicios de red avanzada."

[80] B. Smartekh, "9 recomendaciones y parches para protegerte de wannacry," 5 2017. [En línea]. Disponible: https://blog.smartekh.com/wannacry-9-recomendaciones-parches

[81] "How threat actors are using SMB vulnerabilities - Malwarebytes Labs | Malwarebytes Labs." [En línea]. Disponible: https://blog.malwarebytes.com/101/2018/12/how-threat-actors-are-using-smb-vulnerabilities/

[82] Redhat.com, "CVE-2018-15473- Red Hat Customer Portal." [En línea]. Disponible: https://access.redhat.com/security/cve/cve-2018-15473

[83] Tenable, "ProFTPD mod_copy Information Disclosure | Tenable®." [En línea]. Disponible: https://www.tenable.com/plugins/nessus/84215

[84] Incibe-cert, "CVE-2012-2122 | INCIBE-CERT," 2012. [En línea]. Disponible: https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2012-2122

[85] Tenable, "MySQL 8.0.x < 8.0.23 Multiple Vulnerabilities (Jan 2021 CPU) | Tenable®." [En línea]. Disponible: https://www.tenable.com/plugins/nessus/145251

[86] ——, "UnrealIRCd Backdoor Detection | Tenable®." [En línea]. Disponible: https://www.tenable.com/plugins/nessus/46882

[87] "NVD - CVE-2015-1158." [En línea]. Disponible: https://nvd.nist.gov/vuln/detail/CVE-2015-1158

[88] Redhat.com, "CVE-2015-1158- Red Hat Customer Portal." [En línea]. Disponible: https://access.redhat.com/security/cve/cve-2015-1158

[89] "CVE-2018-6553 : The CUPS AppArmor profile incorrectly confined the dnssd." [En línea]. Disponible: https://www.cvedetails.com/cve/CVE-2018-6553/

[90] Vuldb, "CVE-2018-6553 | Ubuntu Linux CUPS AppArmor access control (111014 / 197201)." [En línea]. Disponible: https://vuldb.com/?id.122780

[91] "CVE-2014-0226 Race condition in the mod_status." [En línea]. Disponible: https://vulmon.com/vulnerabilitydetails?qid=CVE-2014-0226{&}scoretype=cvssv2

[92] Tenable, "Apache 2.4.x < 2.4.10 Multiple Vulnerabilities | Tenable®." [En línea]. Disponible: https://www.tenable.com/plugins/nessus/76622

[93] Redhat.com, "CVE-2015-5600- Red Hat Customer Portal." [En línea]. Disponible: https://access.redhat.com/security/cve/cve-2015-5600

[94] WeLiveSecurity, "¿en qué consiste la vulnerabilidad cross site request forgery (csrf)?" [En línea]. Disponible: https://www.welivesecurity.com/la-es/2015/04/21/vulnerabilidad-cross-site-request-forger-csrf/

[95] Gb-advisors, "¿Cómo prevenir vulnerabilidades de sitios cruzados (CSRF)?" [En línea]. Disponible: https://www.gb-advisors.com/es/como-prevenir-vulnerabilidades-sitios-cruzados-csrf/

[96] Netsparker, "What is SQL Injection & How to Prevent it | Netsparker." [En línea]. Disponible: https://www.netsparker.com/blog/web-security/sql-injection-vulnerability/

[97] Rapid 7, "Apache HTTPD: mod_status buffer overflow (CVE-2014-0226)." [En línea]. Disponible: https://www.rapid7.com/db/vulnerabilities/apache-httpd-cve-2014-0226/

[98] N. vulnerability database, "Nvd - cve-2021-32785," 2 2021. [En línea]. Disponible: https://nvd.nist.gov/vuln/detail/CVE-2021-32785

[99] Vuldb, "Cve-2021-32028 | postgresql conflict divulgación de información." [En línea]. Disponible: https://vuldb.com/es/?id.182161

[100] Solarwinds, "LEM vulnerability, how to solve it? - Forum - Security Event Manager (SEM) - THWACK." [En línea]. Disponible: https://thwack.solarwinds.com/product-forums/security-event-manager-sem/f/forum/31477/lem-vulnerability-how-to-solve-it/43729

[101] S. Martin, "New phpmyadmin packages fix several vulnerabilities," 11 2005. [En línea]. Disponible: https://marc.info/?l=bugtraq&m=113095367912325&w=2

[102] Tenable, "ASP.NET DEBUG Method Enabled | Tenable®." [En línea]. Disponible: https://www.tenable.com/plugins/was/112353

[103] Microsoft, "Disable debugging for asp.net application - asp.net | microsoft docs." [En línea]. Disponible: https://docs.microsoft.com/en-US/troubleshoot/aspnet/disable-debugging-application