INYECCIÓN DE SQL, TIPOS DE ATAQUES Y PREVENCIÓN EN ASP.NET - C#

Orlando Tovar Valencia
Universidad Piloto de Colombia
orlandotovarvalencia@gmail.com
Bogotá, Colombia

Resumen - Estudios revelan que 75% de los ataques ocurren en el nivel de aplicación [1] y que de un 100% de aplicaciones auditadas el 96% contenían vulnerabilidades de seguridad y que de ese porcentaje el 13% fueron atacadas con técnicas de Sql Injection [2]. OWASP (Open Web Application Security Project), en su top 10 de vulnerabilidades del año 2013 y 2010[3] muestra como en orden de ataques la inyección de Sql se encuentra en el primer lugar con más vulnerabilidades explotadas.

Las aplicaciones web se han convertido en un blanco fácil de la infraestructura de TI. Con tantas vulnerabilidades para elegir, los atacantes pueden fácilmente acceder a los datos cada vez más expuestos para ser vulnerados. Este artículo se centra en la explotación de vulnerabilidades en aplicaciones web bajo las técnicas de inyección de Sql, se exponen herramientas, frameworks y buenas prácticas para la prevención de este tipo de ataques, bajo la plataforma Asp.Net con su lenguaje Csharp, ya que el uso de la plataforma Asp.Net y sus lenguajes de programación aportan un 11.8% según el ranking de lenguajes utilizados por Tiobe Software a noviembre de 2014[4].

Abstract - Studies show that 75% of attacks occur at the application level [1] and 100% of audited applications 96% contained security vulnerabilities and that percentage 13% were attacked with techniques Sql Injection [2]. OWASP (Open Web Application Security Project) in its top 10 vulnerabilities in 2013 and 2010 [3] shows that in order SQL injection attacks is in first place with most exploited vulnerabilities.

Web applications have become an easy target for the IT infrastructure. With so many to choose vulnerabilities, attackers can easily access increasingly exposed to be violated data. This article focuses on the exploitation of vulnerabilities in web applications under Sql injection techniques, tools, frameworks and best practices for preventing such attacks are discussed under the Asp.Net platform Csharp language as using the Asp.Net platform and programming languages provide 11.8% according to the ranking of languages used by Tiobe Software to November 2014 [4].

Palabras claves - Lenguaje Sql, Asp.Net, Csharp, Sqlmap, Nessus, Acunetix, Sql Injection, Linq, Entity Framework, Orms, Frameworks.

I. INTRODUCCIÓN

En la actualidad las aplicaciones web se han vuelto indispensables para el manejo de la información en una organización, convirtiéndose en una herramienta que permite al usuario acceder y utilizar un sistema informático a través de internet mediante un navegador, permitiendo el acceso a la información desde cualquier parte del mundo sin importar el sistema operativo que se utilice, solo con acceso a un navegador web y una conexión a internet.

Estas arquitecturas web manejan diferentes tipos de información como pueden ser financieros, organizacionales o de uso general. Las empresas han adoptado este tipo de diseño por el acceso inmediato a sus recursos informativos, obteniendo información relacionada con los clientes, proveedores, usuarios etc. esto hace que el portal "página web" o herramienta "software contable, financiero, etc." se vuelva atractivo a cualquier usuario malintencionado. Según la organización Gartner Inc. 75% de los ataques ocurren a nivel de aplicación o servicio [1], y un estudio realizado por la empresa Cenzic en su reporte "Application vulnerability trends report 2014" [2] demostró que de un 100% de aplicaciones auditadas el 96% contenían vulnerabilidades de seguridad y que de ese porcentaje el 13% fueron atacadas con técnicas de Sql inyección.

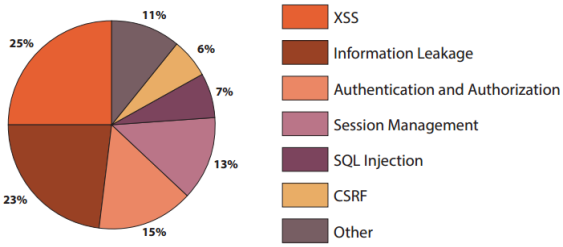

Fig. 1 Tendencias de vulnerabilidades Web 2014. Fuente: Cenzic Inc, http://goo.gl/7rqJvV

Una de las organizaciones más respetadas a nivel mundial en el desarrollo de software en ambiente web OWASP (Open Web Application Security Project), en su top 10 de vulnerabilidades del año 2013 y 2010 muestran como en orden de ataques la inyección de Sql se encuentra en el primer lugar con más vulnerabilidades explotadas[3].

OWASP Top 10 – 2010 (Previo)	OWASP Top 10 – 2013 (Nuevo)
A1 – Inyección	A1 – Inyección
A3 – Pérdida de Autenticación y Gestión de Sesiones	A2 – Pérdida de Autenticación y Gestión de Sesiones
A2 – Secuencia de Comandos en Sitios Cruzados (XSS)	A3 – Secuencia de Comandos en Sitios Cruzados (XSS)
A4 – Referencia Directa Insegura a Objetos	A4 – Referencia Directa Insegura a Objetos
A6 – Defectuosa Configuración de Seguridad	A5 – Configuración de Seguridad Incorrecta
A7 – Almacenamiento Criptográfico Inseguro – Fusionada A9→	A6 – Exposición de Datos Sensibles
A8 – Falla de Restricción de Acceso a URL – Ampliada en →	A7 – Ausencia de Control de Acceso a las Funciones
A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)
<dentro de A6: – Defectuosa Configuración de Seguridad>	A9 – Uso de Componentes con Vulnerabilidades Conocidas
A10 – Redirecciones y reenvíos no validados	A10 – Redirecciones y reenvíos no validados
A9 – Protección Insuficiente en la Capa de Transporte	Fusionada con 2010-A7 en la nueva 2013-A6

Fig. 2 Top 10 vulnerabilidades Web OWASP 2013. Fuente: Proyecto OWASP http://goo.gl/SbZXVl

Las aplicaciones web se han convertido en un blanco fácil de la infraestructura de TI. Con tantas vulnerabilidades para elegir, el atacante puede fácilmente acceder a los datos cada vez más expuestos para ser vulnerados.

Este artículo se centra en la explotación de vulnerabilidades en aplicaciones web bajo las técnicas de inyección de Sql, exponiendo herramientas, frameworks y buenas prácticas para prevenir este tipo de ataques bajo la plataforma Asp.Net y su lenguaje de programación Csharp "C#", ya que el uso de la plataforma Asp.Net y sus lenguajes de programación aportan un 11.8% de uso en desarrollo web según la clasificación de lenguajes utilizados por Tiobe Software a Noviembre de 2014[4].

A. *Arquitectura de una aplicación Web*

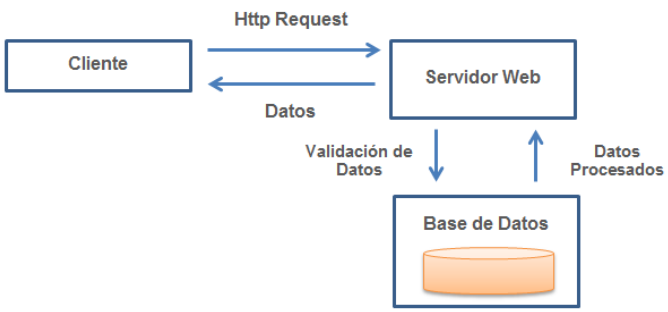

Fig. 3 Arquitectura Web, Fuente: http://www.andromeda-project.org/webarchitectureoverview.html

El cliente envía una solicitud HTTP al servidor Web, esta solicitud tendrá los datos de entrada del usuario. Estos datos serán enviados a la capa de base de datos para su procesamiento por el servidor web que contiene la aplicación contable, financiera etc. Al final la base de datos procesara las consultas y el resultado será enviado al servidor web para que el cliente pueda ver la información procesada. Por lo general el servidor de base de datos contiene muchas bases de datos, y a su vez, cada base de datos contiene tablas que pueden variar en cantidad dependiendo del tamaño de la aplicación, es por esta razón que la base de datos está bajo enorme amenaza por los atacantes.

B. *Base de Datos*

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos relacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un sistema gestor de base de datos es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información.

La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos [5].

C. *Asp.Net - Chsarp*

Asp.Net es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). Asp.Net está construido sobre el Common Language Runtime, permitiendo

a los programadores escribir código Asp.Net usando cualquier lenguaje admitido por el .NET Framework.[6], entre los lenguajes más utilizados para el desarrollo de aplicaciones web bajo este framework se encuentran Csharp "C#" y Visual Basic "VB", ambos creador por Microsoft para estas plataformas.

II. SQL INYECCIÓN (SQL INJECTION)

La inyección de Sql es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar consultas a una base de datos. El origen de la vulnerabilidad radica en el incorrecto chequeo y/o filtrado de las variables utilizadas en un programa que contiene, o bien genera, código Sql. Es, de hecho, un error de una clase más general de vulnerabilidades que puede ocurrir en cualquier lenguaje de programación o script que esté embebido dentro de otro. Se conoce como inyección de Sql, indistintamente, al tipo de vulnerabilidad, al método de infiltración, al hecho de incrustar código Sql intruso y a la porción de código incrustado [7].

A. ¿Cómo es un ataque de inyección Sql?

La inyección de Sql es un tipo de vulnerabilidad en el que un atacante es capaz de insertar scripts o comandos Sql a la base de datos, el cual es ejecutado por una aplicación, dejando al descubierto la base de datos. Los ataques de inyección de Sql pueden ocurrir cuando una aplicación web utiliza los datos suministrados por el usuario sin la validación adecuada o codificación como parte de una consulta. La inyección de Sql permite a un atacante crear, leer, actualizar, modificar o eliminar los datos almacenados en la base de datos. Con un tipo de vulnerabilidad el atacante puede tener acceso a información confidencial, como puede ser el número de tarjeta de crédito, contraseñas, datos financieros o cualquier tipo de información contenida en la base de datos.

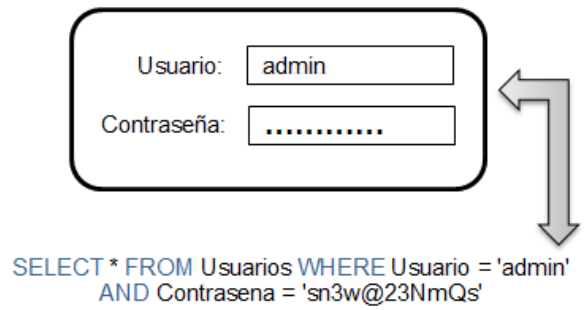

Fig. 4 Ejemplo Consulta Sql, Fuente: Autor

En la figura No 4 se observa cómo funciona un sistema de ingreso "login" a una aplicación normal, se envían dos "variables", usuario y contraseña. Las cuales son incrustadas en la sentencia Sql que retorna el registro encontrado en la tabla "Usuarios", este es el método más fácil de verificar las credenciales de un usuario y utilizado para ingreso a las aplicaciones.

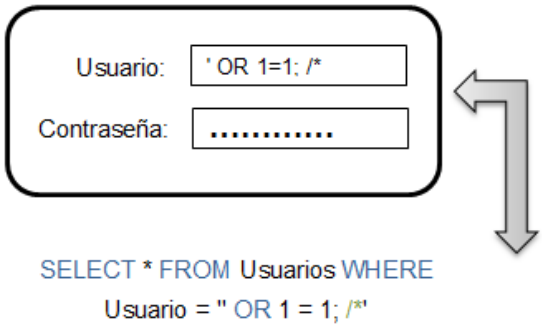

Fig. 5 Ejemplo de Sql Inyección, Fuente: Autor

En la figura No 5, muestra cómo se puede explotar fácilmente un sistema de ingreso "login" con un comando de Sql sencillo interrumpiendo la cadena en la que va la instrucción Sql para ser ejecutada en la base de datos.

Algunos de los conceptos claves de la inyección de Sql son:

- La inyección de Sql es una vulnerabilidad de software que se produce cuando los datos introducidos por los usuarios se envían a la base de datos como parte de una consulta Sql.
- Los atacantes diseñan especialmente datos de entrada al intérprete de Sql para engañar al intérprete y ejecutar comandos no deseados.
- La inyección de Sql explota las vulnerabilidades de seguridad en la capa de base de datos. Una vez explotado la

vulnerabilidad el atacante pueden crear, leer, modificar o eliminar datos sensibles.

III. TIPOS DE ATAQUES

Existen diferentes métodos para explotar este tipo de vulnerabilidades, según sea el objetivo del atacante, se realizan ataques en conjunto o secuencialmente. A continuación se muestra una posible clasificación de ataques de Sql inyección algunos basados en las técnicas de explotación de Owasp [8].

A. Introducir una instrucción siempre verdadera "True"

Este tipo de ataque inyecta Sql a la instrucción de una consulta condicional para que al ser ejecutado siempre sea cierto. Este ataque se utiliza para omitir el control de autenticación y generar acceso a los datos mediante la explotación vulnerable de un campo de entrada, por ejemplo una caja de texto, para esta vulnerabilidad se utiliza cláusula WHERE.

```
SELECT * FROM Usuarios WHERE Usuario = 'admin' AND Contrasena = '23VbsNpl' OR '1' = '1
```

La declaración (1 = 1) siempre va a hacer verdadera por consiguiente la consulta siempre va a generar un resultado positivo.

B. Lógica incorrecta en las consultas Sql

Cuando una consulta es rechazada por el motor de base de datos, un mensaje de error es devuelto incluyendo información para que su depuración sea fácil o útil. Estos mensajes de error ayuda a que el atacante pueda encontrar vulnerabilidad en la aplicación y en la secuencia de comandos que se insertan a la base de datos. Una forma muy fácil para encontrar este tipo de vulnerabilidad es inyectando entradas "basura" para producir el error de sintaxis. En este ejemplo el atacante induce a generar un error de coincidencia de tipos al inyectar el siguiente texto en el campo de entrada.

1) Url con parámetros:
   www.ejemplo.com/usuarios.aspx?id=888
2) Url con inyección de Sql carácter delimitador de cadena de datos "'" introducido:
   www.ejemplo.com/usuarios.aspx?id=888'
3) El mensaje de error se muestra por la interrupción del valor enviado:
   SELECT * FROM Usuarios WHERE id =8864\';

Desde un mensaje de error se puede obtener por ejemplo el nombre de la tabla y los campos. Este tipo de ataque es uno de los más fáciles de explotar.

C. Unión de consultas.

Mediante esta técnica los atacantes unen, una o varias consultas por medio de la palabra "UNION", con la cual se puede obtener datos sobre otras tablas de la aplicación. Por ejemplo la siguiente consulta ejecutada desde el servidor:

1) SELECT Nombre, Telefono FROM Usuarios WHERE id = id
2) Al inyectar el siguiente valor:
   id = 1 UNION ALL SELECT NumeroTarjeta, 1, 1 FROM TarjetasCredito
3) Generando la siguiente consulta:
   SELECT Nombre, Telefono FROM Usuarios WHERE id = 1 UNION ALL SELECT NumeroTarjeta, 1 FROM TarjetasCredito

La segunda consulta se unirá al resultado de la consulta original obteniendo todos los números de las tarjetas de crédito.

D. Ejecutando consultas adicionales.

Con este tipo de vulnerabilidad, el atacante explota la base de datos por el delimitador de consulta, tal como ";", al anexar una consulta adicional a la consulta original. Con este tipo de ataque se puede ejecutar una o varias consultas distintas. Normalmente, la primera consulta es la consulta genuina o correcta, mientras que las consultas siguientes podrían ser las consultas malintencionadas. Así el atacante puede inyectar cualquier comando Sql a la base de datos. En el siguiente ejemplo, el atacante desea eliminar la tabla de usuarios, para esto inyecta después de la instrucción "0;" un DROP TABLE Usuarios. La consulta Sql completa seria la siguiente:

```
SELECT FROM Usuarios WHERE Usuario = 'admin' AND Contrasena = 'xc#12Ns'; DROP TABLE Usuarios;
```

Esta falencia ocurre debido a que el carácter ";" en la base de datos acepta ambas consultas y los ejecuta una después de la otra.

E. Procedimientos almacenados.

Un procedimiento almacenado es una parte de la base de datos en la cual el desarrollador podría establecer una abstracción adicional "consultar, actualizar, eliminar, etc." en la capa de base de datos. El procedimiento almacenado podría ser codificada por el programador dejando vulnerabilidades al no tener el conocimiento necesario para una estructura correcta y segura, por lo que se podría inyectar Sql desde los formularios de la aplicación web. Dependiendo de la estructura del procedimiento almacenado en la base de datos hay diferentes maneras de explotarlos. En el siguiente ejemplo, el atacante vulnera el procedimiento almacenado por medio de los parámetros enviados.

```
CREATE PROCEDURE Autenticar DECLARE @Usuario varchar(100), DECLARE @Contrasena Varchar(32), AS
EXEC("SELECT * FROM Usuarios WHERE Usuario = ' + @Usuario + ' AND Contrasena = ' + @Contrasena + '");
GO
```

El procedimiento devuelve verdadero o falso. Al enviar al procedimiento el valor ya sea para el campo Usuario o Contraseña, la palabra reservada "SHUTDOWN; --". Se genera la siguiente consulta:

```
SELECT * FROM Usuarios WHERE Usuario = 'admin' AND Contrasena = ''; SHUTDOWN; --
```

La primera búsqueda se ejecuta por el contrario la segunda consulta es la mal intencionada, forjando a que la base de datos se apague. Es necesario tomar medidas ya que procedimientos almacenados son tan vulnerables como el código de una aplicación web.

F. Interferencia

Este tipo de ataque cambia el comportamiento de la base de datos o aplicación. Hay dos técnicas para este tipo de ataque que se basan en inferencia: la inyección a ciegas y temporización de ataques.

1) Inyección a ciegas: A veces los desarrolladores esconden los detalles de error los cuales ayudan a los atacantes a comprometer la base de datos, con lo cual le es más difícil encontrar una vulnerabilidad de este tipo, pero no lo hace imposible. La técnica del atacante es comenzar a generar consultas solo con posibles respuestas verdadera o falsa. Por ejemplo inyecciones en el campo de inicio de sesión con dos posibles instrucciones Sql.

```
SELECT * FROM Usuarios WHERE Usuario= 'admin' AND 1 = 0 -- AND Contrasena = ''
```

```
SELECT * FROM Usuarios WHERE Usuario = 'admin' AND 1 = 1 -- AND Contrasena = ''
```

Si se asegura la aplicación y se toman medidas para los ataques de Sql inyección, ambas consultas se ejecutarían sin éxito, debido a la validación de entrada. Pero si no hay ningún tipo de validación de entrada, el atacante puede intentar enviar las consultas. Primero envía la consulta inicial y recibe un mensaje de error debido a la instrucción "1 = 0". Así que el atacante puede obtener información relacionada con el error. A continuación, el atacante envía la segunda consulta que siempre genera una instrucción verdadera. Si no hay ningún error de inicio de sesión, entonces el atacante encuentra el campo de inicio de sesión con vulnerabilidad a la inyección de Sql.

2) Temporización de ataques: Un ataque de sincronización permite a un atacante reunir información de una base de datos mediante la observación de los retrasos de tiempo en las respuestas de la base de datos. Esta técnica es explotada mediante el uso de la instrucción "if-then" la cual hace que el motor de base de datos ejecute el Sql con una declaración de retardo de tiempo dependiendo de la lógica que inyecta. Este ataque es similar a los ataques de inyección a ciegas el atacante puede medir el tiempo en que por ejemplo la página tarda en cargar para determinada instrucción inyectada. Esta técnica utiliza una combinación de preguntas con la sentencia "if" para la inyección en las consultas. WAITFOR es una palabra clave del motor de base de datos, que hace que la base de datos retrase su respuesta por un tiempo determinado. Por ejemplo, en la siguiente consulta:

DECLARE @BaseDatos varchar(8000) SELECT @BaseDatos = db_nameO IF (ascii(substring(@BaseDatos, 1, 1)) & (power(2, 0))) > 0 WAITFOR DELAY 0:0:5

La base de datos hará una pausa por cinco segundos si el primer bit del primer byte del nombre de la base de datos es mayor a 1. Esto genera un retraso en tiempo de respuesta cuando la condición es verdadera. También el atacante puede pedir una serie de preguntas sobre la base de datos sacando información importante sobre la misma para generar un ataque más agresivo.

IV. ¿CÓMO SABER SI SE ES VULNERABLE?

La mayor parte de las técnicas existentes, como el filtrado, análisis de flujo de información, pruebas de penetración, y codificación defensiva, puede detectar y prevenir un subconjunto de las vulnerabilidades que llevan a taques de Sql inyección. Existen diferentes herramientas de software que generan análisis a gran escala para encontrar vulnerabilidades de inyección de Sql. A continuación se describen algunas que pueden ser muy útil al momento de saber si la aplicación que se tiene en la organización o que se esta desarrollando es vulnerable o no a este tipo de ataques.

A. Acunetix

Acunetix es un escáner de vulnerabilidades de aplicaciones web. La herramienta está diseñada para encontrar agujeros de seguridad en las aplicaciones web de la organización que un atacante podría aprovechar para obtener acceso a los sistemas y datos. Acunetix comprueba los sistemas en busca de múltiples vulnerabilidades incluyendo:

- SQL injection
- Cross Site Scripting
- Passwords débiles

Acunetix puede utilizarse para realizar escaneos de vulnerabilidades en aplicaciones web y para ejecutar pruebas de acceso frente a los problemas identificados. La herramienta provee sugerencias para mitigar las vulnerabilidades identificadas y puede utilizarse para incrementar la seguridad de servidores web o de las aplicaciones que se analizan [9].

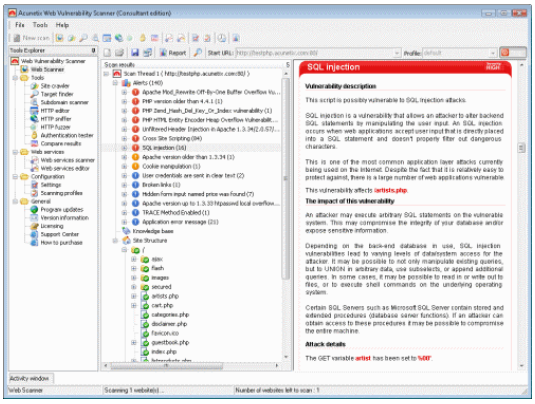

Fig. 6 Reporte de vulnerabilidades Acunetix, Fuente: Acunetix, http://www.acunetix.com/

B. Nessus.

Nessus es un programa de escaneo de vulnerabilidades en diversos sistemas operativos. Consiste en un daemon, nessusd, que realiza el escaneo en el sistema objetivo, y Nessus, el cliente (basado en consola o gráfico) que muestra el avance e informa sobre el estado de los escaneos. Desde consola Nessus puede ser programado para hacer escaneos programados con cron [10].

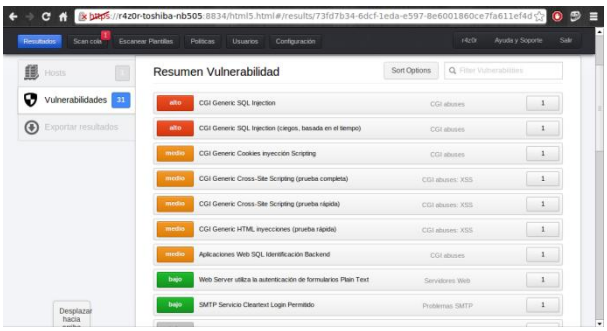

Fig. 7 Reporte de vulnerabilidades Nessus, Fuente: Nessus - Tenable Network Security, http://www.tenable.com/

Entre sus características Nessus escanea de forma exhaustiva si el aplicativo web contiene vulnerabilidades de inyección de Sql como se muestra el reporte de vulnerabilidades de la figura No 7.

C. Sqlmap.

Sqlmap es una herramienta desarrollada en python para realizar inyección de código sql automáticamente. Su objetivo es detectar y aprovechar las vulnerabilidades de inyección Sql en aplicaciones web. Una vez que se detecta una o más inyecciones Sql en el host de destino, el usuario puede elegir entre una variedad de opciones entre

ellas, enumerar los usuarios, los hashes de contraseñas, los privilegios, las bases de datos, o todo el volcado de tablas / columnas específicas del DBMS [11].

Fig. 8 Consola de comandos Sqlmap, Fuente: Backtrack Linux Org. http://www.backtrack-linux.org/screenshots/

D. Pruebas unitarias.

Una parte fundamental para saber si la aplicación que se está auditando o se está desarrollando es vulnerable a este tipo de ataque, es generando pruebas unitarias, probando con simple instrucciones de Sql en datos de entrada a la aplicación por ejemplo con un carácter tan sencillo como “ ' ” en un campo de una caja de texto o en las variables que son pasadas como parámetros entre páginas:

http://ejemplo.com/buscar.aspx?id=' OR '1'='1

Por ejemplo al cambiar el parámetro de “id” cambia el significado de la consulta regresando todos los registros de la tabla que esté generando la búsqueda.

V. PREVENIR ATAQUES DE INYECCIÓN DE SQL EN ASP.NET – C#

El principal problema con estos ataques es que si se deja que el usuario del programa introduzca libremente caracteres sin control ninguno (mediante formularios, por ejemplo) puede llegar a aprovecharse de estos errores de programación para crear ataques maliciosos sobre la aplicación.

Prevenirlas no ha sido fácil, la gran mayoría de programadores han querido filtrar los caracteres que reciben antes de crear las consultas, y en ese caso suceden dos cosas: O no siempre se filtran todos los caracteres perjudiciales que pudieran intervenir en una inyección de código Sql, o se filtran algunos caracteres que pudieran ser de uso común, por lo que se le restringe la usabilidad al usuario y de la aplicación. Las soluciones definitivas y conocidas para este tipo de ataques son usar métodos que no puedan confundir parámetros con comandos, además del surgimiento de Frameworks y Orms para el trabajo con la base de datos, a continuación se listara una de las posibles soluciones para evitar este tipo de ataques.

A. Convertir siempre el valor a su tipo correspondiente.

Si por ejemplo se está esperando un valor de tipo numérico, se debería intentar “parsear” o convertir este valor a dicho tipo para asegurarse de que no incluye texto adicional o malintencionado:

```
var id = Request.QueryString["id"];
int idConvertido;
Int32.TryParse(id, out idConvertido);
```

Fig. 9 Convirtiendo el valor a un dato entero, Fuente: Autor.

B. Parametrizar las consultas Sql.

Una de las normas más importantes a la hora de desarrollar y utilizar consultas a la base de datos donde se tiene que generar “SELECT, INSERT, UPDATE” es utilizar parámetros para tal fin.

```
SqlConnection con = new SqlConnection(cadenaConexion);
SqlCommand cmd = new SqlCommand("SELECT "
                                + " * "
                               +" FROM "
                                 +" Usuarios " +
                               +" WHERE "
                                    +" Usuario = @varUsuario AND "
                                    +" Password= @varPassword", con);

/* Convertimos en literal estos parámetros, por lo que no podrán
hacer la inyección **/

cmd.Parameters.Add("@varUsuario", SqlDbType.VarChar, 32).Value = varUsuario;
cmd.Parameters.Add("@varPassword", SqlDbType.VarChar, 64).Value = varPassword;
```

Fig. 10 Generando consultas por medio de parámetros, Fuente: Autor.

En .Net se evitara la inyección en Sql para el ejemplo con CSharp “C#” como lenguaje de programación estableciendo el tipo de parámetro como literal con SqlDbType.VarChar, para este caso una variable de tipo String o alfa numérica.

Otra forma como CSharp maneja el paso de parámetros para las consultas es con la palabra reservada “AddWithValue” que es una forma muy similar a la anterior.

```
using( SqlConnection con = (acquire connection) ) {
    con. Open();
    using( SqlCommand cmd = new SqlCommand("SELECT "
                                + " * "
                               +" FROM "
                                +" Usuarios " +
                               +" WHERE "
                                    +" Usuario = @varUsuario AND "
                                    +" Password= @varPassword", con) ) {
        /* Convertimos también en literales los parámetros */
        cmd.Parameters.AddWithValue("@varUsuario", user);
        cmd.Parameters.AddWithValue("@varPassword", password);
        using( SqlDataReader rdr = cmd.ExecuteReader() ){
            /* [...] */
        }
    }
}
```

Fig. 11 Generando consultas por medio de parámetros, Fuente: Autor.

C. Usar una cuenta con permisos restringidos a la base de datos.

Un dato importante a tener en cuenta es asegurarse de que la cuenta de usuario utilizada por la aplicación tiene los permisos necesarios para poder acceder y/o modificar unos datos concretos pero también que sea lo suficiente restrictiva para no alterar otro tipo de datos [12].

D. No mostrar al usuario la información de error generada por la base de datos.

Los mensajes de error pueden ser lo suficientemente descriptivos como para que el atacante obtenga información acerca de la base de datos, se hace necesario controlar este tipo de vulnerabilidad, para esto CSharp cuenta con la instrucción "try catch" como se muestra en la figura No 12.

```
try{
    /* ..... */
} catch(SqlException e)
{
    /* ..... */
}
```

Fig. 12 Ocultando errores de Sql, Fuente: Autor.

La idea es evitar que el error sea visto por el usuario final de la aplicación.

E. Rechazar las peticiones con caracteres sospechosos.

Al igual que muchos lenguajes de programación, el lenguaje Sql contiene muchos elementos que son utilizados para generar consultas en las bases de datos. Aunque generar filtrado de caracteres no sea cien por ciento recomendable ya que esto no implica que sea totalmente preventivo en la práctica puede "prevenir" este tipo de ataque.

Carácter especial	Significado SQL
;	Delimitador de consultas.
'	Carácter delimitador de cadena de datos.
-	Comentario.
/* */	Delimitadores de comentario. El texto entre /* y */ no es evaluado.
xp_	Se utiliza en el inicio del nombre de procedimientos almacenados extendidos de catálogo, como xp_cmdshell.

Fig. 12 Caracteres especiales de Sql, Fuente: Blog de Cheo Redondo, http://goo.gl/N6z6xQ

F. Uso de LINQ (Language-Integrated Query)

Language-Integrated Query (LINQ) es un conjunto de características que agrega capacidades de consulta eficaces a la sintaxis de los lenguajes C# y Visual Basic. LINQ incluye patrones estándar y de fácil aprendizaje para consultar y actualizar datos, y su tecnología se puede extender para utilizar potencialmente cualquier tipo de almacén de datos [13].

LINQ pasa todos los datos a la base de datos a través de parámetros Sql. Así, aunque la consulta Sql se compone de forma dinámica, los valores son sustituidos de lado del servidor a través de parámetros que previenen en contra la causa más común de los ataques de inyección de Sql.

```
List<EntidadProducto> datos = new List<EntidadProducto>();
var serv = (from s in contexto.Productos
            join sl in contexto.Categoria on s.Categoria equals sl.ID
            select s).ToList();
```

Fig. 13 Código generado con Linq, Fuente: Autor.

G. Uso de Entity Framework.

Entity Framework (EF) es un asignador objeto-relacional que permite a los desarrolladores de .Net trabajar con datos relacionales usando objetos específicos del dominio. Elimina la necesidad de la mayor parte del código de acceso a datos que los desarrolladores suelen tener que escribir [14].

A diferencia de las consultas clásicas de Sql Entity Framework están compuestas usando la manipulación de cadenas o concatenación, y no son susceptibles a los ataques tradicionales de inyección Sql.

H. Relacionar la base de datos.

Aunque relacionar la base de datos no influye para evitar un ataque de inyección de Sql, si se puede prevenir por ejemplo que se quieran eliminar o "DELETE" o borrar "DROP" información desde

una tabla, al estar relacionada con otras tablas se "podría" evitar la posible pérdida de registros.

VI. CONCLUSIONES

Los ataques de inyección de Sql son los más comunes y fáciles de explotar, aunque hay herramientas automatizadas para generar ataques de este tipo, explotarlos de manera sencilla es muy fácil y no se necesita gran conocimiento en lenguaje Sql. Los usuarios finales que manejan las aplicaciones, muchas veces por error envían por medio de la aplicación datos con caracteres especiales a la base de datos, la cual al interpretarla genera errores de sintaxis, un usuario normal no entiende lo que la aplicación le responde, pero si esto por ejemplo lo utiliza una persona con mayor conocimiento, puede comenzar a realizar un tipo de ataque como los que se vieron anteriormente, buscando obtener la mayor cantidad de información sobre la estructura de la base de datos que está atacando.

Para prevenir este tipo de ataques también se puede tomar ciertas medidas, como por ejemplo las que se describieron anteriormente, no se necesitan gran cantidad de contra medidas o prevenciones para evitar este tipo de vulnerabilidad, el desarrollador de software al tener conocimientos de seguridad y bien claro que tipo de parámetro va a recibir la base de datos previene la exposición a cualquier tipo de inyección. El uso frameworks, orms como Linq o Entity Framework para automatizar mejor el acceso a datos por parte de la aplicación ayudan de una forma más limpia a prevenir esta vulnerabilidad. En un gran porcentaje las aplicaciones fallan por desconocimiento del grupo de desarrolladores, por no manejar estándares de calidad en el desarrollo de software, o por el poco tiempo invertido en aplicar seguridad en el desarrollo.

Este tipo de ataques una vez explotados tienen un alto grado de impacto en la organización que soporta la aplicación, página web, etc. ya que la integridad, disponibilidad y autenticidad se ven comprometidas seriamente.

REFERENCIAS

[1] - Gartner Groug, Estadísticas sobre seguridad [en línea] disponible en: http://goo.gl/7rqJvV
[2] - Cenzic, Application vulnerability trends report 2014 [en línea] disponible en: http://goo.gl/odW4ee
[3] - OWASP, TOP 10 – 2013 Vulnerabilidades Aplicaciones Web [en línea] disponible en: http://goo.gl/SbZXVl
[4] - Tiobe Software, Estadística de lenguajes de programación, [en línea] disponible en: http://goo.gl/uDRcgR
[5] - Abraham Silberschatz, Henry F. Korth, Fundamentos de bases de datos, Cuarta edición 2002.
[6] - Asp.Net, Enciclopedia Libre, [en línea] disponible en: http://es.wikipedia.org/wiki/ASP.NET
[7] - Sql Inyección, Enciclopedia Libre, [en línea] disponible en: http://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL
[8] - Owasp, Testing for SQL Injection (OTG-INPVAL-005) [en línea] disponible en: http://goo.gl/3og0bZ
[9] - Acunetix, testing for aplications [en línea] disponible en: http://www.acunetix.com/
[10] - Tenable, Nessus scan [en línea] disponible en: http://www.tenable.com/products/nessus
[11] - Dragon Jar, Herramienta Sql Map para inyección de Sql [en línea] disponible en: http://goo.gl/3GY6DB
[12] - Tips para evitar sql Injection, [en línea] disponible en: http://www.returngis.net/2010/10/tips-para-evitar-sql-injection/
[13] - Microsoft, Linq, [en línea] disponible en: http://msdn.microsoft.com/eses/library/vstudio/bb397926(v=vs.110).aspx
[14] - Microsoft, Entity Framework, [en línea] disponible en: http://msdn.microsoft.com/es-co/data/ef.aspx