

Unidad Nro. 7: VULNERABILIDADES

Objetivos Específicos:

Que el estudiante conozca e identifique los principales riesgos en aplicaciones web.
Que el estudiante pueda evaluar la seguridad de aplicaciones web a través de herramientas e interpretar los resultados.

Contenidos:

Prevención de aplicaciones web. Implementación de seguridad a través de filtros.
Inyección de script (XSS - Cross Site Scripting). Inyección de Sql (SQL Injection).
Productos y servicios que evalúan seguridad a nivel de servidores y de aplicaciones.
Recomendaciones.

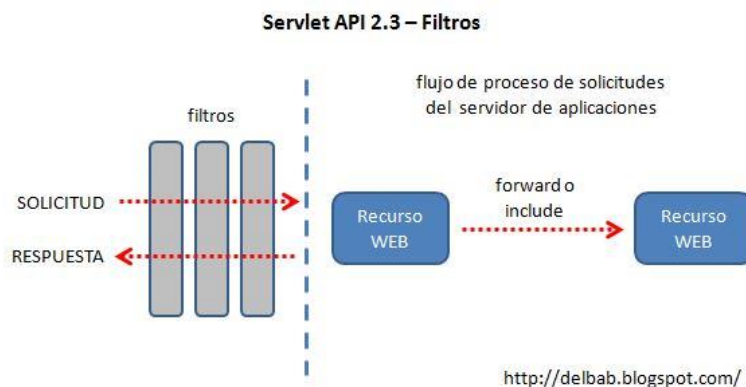
Riesgos o Vulnerabilidades de las aplicaciones Web.

Concepto: Una vulnerabilidad del software es un fallo de seguridad en una aplicación a través de la cual, un atacante puede llegar a comprometer la seguridad de todo el sistema sobre el que se ejecuta esa aplicación.

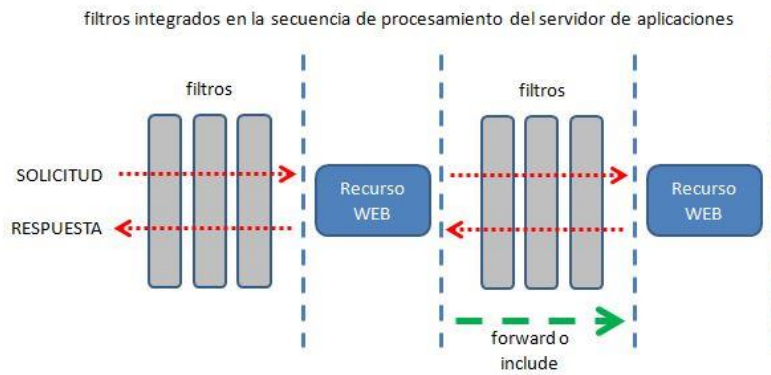
Implementación de seguridad a través de filtros

El servidor web Apache Tomcat provee una infraestructura denominada Filtros que permite interceptar el flujo de entrada y salida desde y hacia una página (.jsp) e intervenirlo. La gracia es que el filtro se programa una vez y permite realizar la tarea en todas las páginas existentes en el contexto.

La infraestructura de filtros está disponible desde la Servlet API 2.3 y fue mejorada en la versión 2.4, permitiendo que los filtros sean aplicados incluso en el traspaso de las solicitudes entre las páginas (por medio de las directivas forward o include). Los diagramas siguientes muestran cómo operan los filtros en el flujo de procesamiento de una página en las dos versiones.



Servlet API 2.4 – Filtros



<http://delbab.blogspot.com/>

2. Ejemplos de Filtros

Considerando el flujo anterior, un Filtro se puede construir para realizar diversas tareas. Algunos ejemplos simples son los siguientes:

Validaciones de Seguridad. Validar que cada página cumpla una condición determinada, por ejemplo, que el usuario esté autenticado.

Compresión de Salidas/Entradas. Comprimir un archivo automáticamente para minimizar el tiempo de descarga.

Registro de Eventos. Generar un log de auditoría respecto con la información de los accesos a las páginas de la aplicación.

Modificación/Intervención de Salidas/Entradas. Eliminación de caracteres especiales, sobrantes, limpieza de HTML, etc.

La implementación de un filtro implica las siguientes actividades:

- Construcción de la clase que implementará el filtro con la funcionalidad deseada.
- Configuración del filtro en Tomcat para que realice su tarea

Obviamente, la construcción del filtro debe realizarse primero que la configuración, sin embargo, voy a describir la configuración primero porque es relevante para entender el "encadenamiento" de los filtros en el procesamiento y la construcción correspondiente.

3. Configuración del Filtro

La configuración del filtro se realiza en el archivo web.xml del contexto (dentro de la carpeta WEB-INF/). Se debe agregar una configuración como la siguiente para cada filtro que se quiera habilitar:

```
<filter>
  <filter-name>NOMBRE</filter-name>
  <filter-class>CLASE</filter-class>
  <init-param>
    <param-name>PARAM_1</param-name>
```

```

    <param-value>VALOR_1</param-value>
  </init-param>
  :
  <init-param>
    <param-name>PARAM_n</param-name>
    <param-value>VALOR_n</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>NOMBRE</filter-name>
  <url-pattern>PATRON</url-pattern>
</filter-mapping>

```

3.1 Sección <filter>.

Permite configurar un filtro en el contexto.

NOMBRE. Corresponde a un nombre lógico que identifica la acción/objetivo que realiza el filtro. Por ejemplo: LoginFilter, RegExFilter, HitCountFilter, etc.

CLASE. Corresponde al nombre de la clase que implementa el filtro. Obviamente, debe ser una clase que esté disponible para el contexto de la aplicación.

PARAM_i y VALOR_i. Se pueden agregar parámetros de configuración relevantes para el filtro agregando nodos del tipo <init-param>. Cada nodo contiene un nombre (PARAM_i) y un valor (VALOR_i).

3.2 Sección <filter-mapping>.

Permite configurar a qué páginas se aplicará el filtro.

NOMBRE. Corresponde al nombre lógico del filtro que se indicó en la sección anterior. Debe ser el mismo para que aplique correctamente.

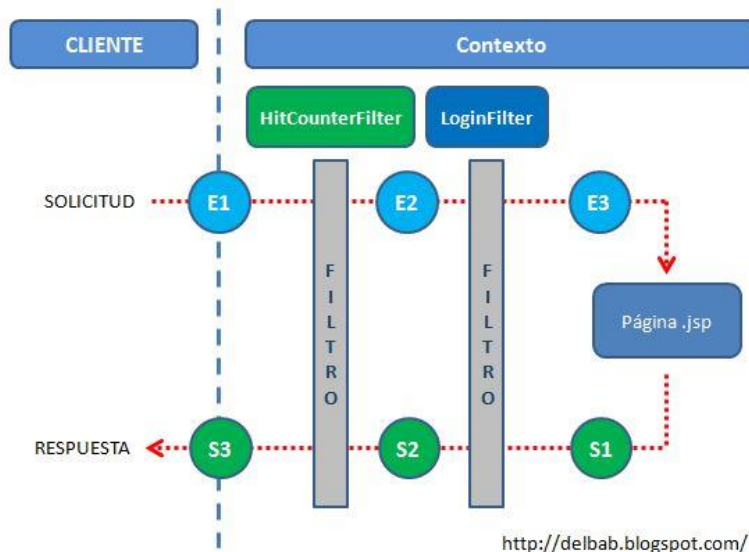
PATRON. Corresponde al patrón de páginas para las cuales se desea que se aplique el filtro. Un ejemplo es /* que indica que aplica a todas las páginas del contexto.

Un aspecto importante de la configuración en el archivo web.xml es que el orden en el que se declaran los filtros. Por ejemplo, supongamos que tenemos dos filtros: HitCountFilter y LoginFilter. El primero registra el número de accesos a las páginas del sitio y el segundo valida que un usuario esté autenticado. Veamos dos configuraciones posibles en el web.xml:

Configuración a)	Configuración b)
<pre> <filter> <filter-name>HitCounterFilter</...> : </filter> <filter> <filter-name>LoginFilter</...> : </filter> </pre>	<pre> <filter> <filter-name>LoginFilter</...> : </filter> <filter> <filter-name>HitCounterFilter</...> : </filter> </pre>

La configuración a) implica que primero se registrarán los accesos y luego se validará al usuario. La configuración b) implica que primero se validará al usuario y luego se registrarán los accesos.

Lo anterior se denomina encadenamiento de filtros (Filter Chain) y establece el orden en el que se deben aplicar los filtros. Lo importante de esto es que los filtros se aplican en cadena sobre la entrada y, luego, de manera inversa sobre la salida. Considerando el ejemplo a) anterior, se produce la siguiente secuencia:



La entrada (E1) es recibida y procesada por HitCounter. El resultado (E2) es entregado a LoginFilter quien la procesa. El resultado (E3) es entregado a la página.jsp para su procesamiento. A esta altura, la ejecución de la página.jsp es similar a la ejecución que se realizaría si no hubiera filtros configurados, es decir, la página.jsp se programa como una página normal, independiente de los filtros que podrían haber sido configurados previamente. El flujo de salida opera de la misma manera permitiendo realizar acciones sobre el resultado (output) de la página.jsp. Ejemplos de acciones posibles a este nivel son la compresión del HTML (eliminando espacios en blanco) o el reemplazo de contenido por otro utilizando expresiones regulares.

4. Construcción del Filtro

Un filtro se construye heredando de la clase `javax.servlet.Filter` y, en él, se deben implementar tres métodos:

`init(FilterConfig filterConfig).`

Este método se ejecuta cuando el contexto es cargado por el servidor y sólo se ejecuta una vez. El parámetro `filterConfig` permite acceder a las configuraciones que se hayan realizado para el filtro en el archivo `web.xml` por medio de la secciones `<init-param/>`.

`doFilter(ServletRequest request, ServletResponse response, FilterChain chain)`. Este método se ejecuta cuando se requiere aplicar el filtro a la entrada (`request`) y a la salida (`response`). El parámetro `chain` es el que permite traspasar el control al siguiente filtro en la cadena. En casos específicos, se puede interrumpir el flujo programáticamente. Por ejemplo, si el filtro `LoginFilter` identifica que el usuario no está autenticado, no es necesario ejecutar la página `.jsp` correspondiente.

`destroy()`. Este método se ejecuta cuando se baja el contexto y puede/debe ser utilizado para liberar recursos que se hayan utilizado para la operación del filtro. Por ejemplo, cachés, contadores, etc.

Inyección de Script (XSS – Cross Site Scripting)

Se trata de una de las vulnerabilidades más comunes en aplicaciones web, también unas de las mas antiguas, sin embargo este tipo de ataques frecuentemente son los mas exitosos y con mayor nivel de evolución dado que con bastante continuidad surgen técnicas cada vez mas elaboradas para conseguir alguna forma de XSS sobre una aplicación Web.

Los ataques XSS se diferencian en varias categorías:

-XSS Reflected

Se trata de un ataque que no se ejecuta en conjunto con la aplicación web, por ejemplo cuando una página en un sitio se carga, esto significa que el contexto de este ataque solamente llega hasta la petición desarrollada por un cliente, normalmente estos ataques suelen asociarse con el robo de cookies, secuestro de sesiones, acceder a historial de la víctima e inclusive acceder a contraseñas almacenadas en el navegador utilizado.

Este tipo de ataque al igual que prácticamente todas las técnicas asociadas con XSS, se encuentran directamente relacionadas con validaciones de los parámetros de entrada que espera una aplicación web. Filtros y validaciones de los datos que un usuario puede ingresar a una aplicación son frecuentemente la razón principal para que se produzcan esta clase de brechas de seguridad.

-XSS Stored

Se trata de una variación del ataque anteriormente mencionado, sin embargo es el mas peligroso, (y obviamente el más deseado por un atacante) dado que el contexto de este ataque no se limita solamente al contexto del navegador web de un usuario, sino que por el contrario puede afectar directamente a todos los usuarios que acceden a la aplicación, por esta razón es una de las vulnerabilidades mas peligrosas. Funciona de un modo similar al anterior, con la diferencia que en este caso, la vulnerabilidad se encuentra almacenada

de forma persistente en la aplicación, ejemplos típicos de este tipo de ataques son foros o sitios donde se pueden incluir comentarios, así como otros tipos de entradas que permite al usuario ingresar texto y estás a su vez, permiten la inclusión de código HTML o JavaScript.

-DOM XSS

Este tipo de ataque se basa en los dos anteriores, la diferencia radica en que se aprovecha la API DOM que tienen los navegadores web para acceder a determinados objetos del navegador, como por ejemplo, funciones en javascript. De esta forma es posible manipular eventos, navegación y otras características que se ejecutan en el lado del cliente. El atacante debe tener buenos conocimientos sobre JavaScript y DOM Api.

-Flash XSS

Consiste en el aprovechamiento de Flash para inyectar código malicioso en etiquetas <object>. En este punto se puede utilizar ActionScript para acceder a las variables de una película hecha en Flash (SWF), también se pueden aprovechar determinadas funciones de Flash para inyectar código malicioso, funciones tales como getURL o loadMovie pueden ser objeto de ataque.

-CSRF

Es tipo de ataque es un poco diferente a los anteriores, consiste en aprovechar la confianza que tiene un sitio en un usuario determinado, si se mira detenidamente los ataques clásicos XSS funcionan justo al contrario, es decir, explotando la confianza que tiene un cliente sobre un sitio web determinado. Consiste en explotar la imposibilidad que tiene una aplicación web en diferenciar entre un usuario legítimo/víctima de un atacante.

Este tipo de ataque habitualmente funciona en tres pasos:

1. El atacante crea un script malicioso hospedado en un servidor web que él controla.
2. La víctima inicia sesión en una página web y mantiene su sesión
3. El atacante envía un enlace al usuario con el script malicioso, el usuario lo “abre” siendo víctima de algún tipo de técnica de ingeniería social (el modo más frecuente). Una vez el script es ejecutado el atacante tendrá la posibilidad de robar cookies y otros objetos del navegador correspondientes a la sesión activa de la víctima.

En realidad en este tipo de ataque, existe una suplantación de la identidad de la víctima y el atacante se hace pasar por alguien que no es.

-XFS

Se trata de una variante de un ataque XSS clásico, consiste en la inyección del código malicioso utilizado por el atacante pero inyectando frames para cargar código externo y evidentemente, sin la autorización de la víctima. Se manipulan las variables de la petición.

-XAS – Cross Agent Scripting

Se trata de inyectar código en una aplicación web por medio de la modificación de las cabeceras HTTP, en este caso, simplemente modificando el parámetro “User-Agent” del navegador web del atacante, y estableciendo como valor el código a inyectar, de esta forma si por ejemplo la aplicación web trata de determinar el User-Agent del navegador, realmente lo que terminará por ejecutar será el código definido en dicho campo, por ejemplo: En lugar de tener el valor “Mozilla” se tiene el valor “<script>alert(document.cookie);</script> ” de esta forma cuando una aplicación intente acceder a dicho campo, realmente se ejecutará este sencillo script.

-XRS – Cross Referer Script

Funciona del mismo modo que XAS, sin embargo en lugar de modificar la cabecera correspondiente al User Agent del navegador, se cambia el valor de la cabecera Referer y de esta forma, se puede inyectar el código malicioso en dicha propiedad.

Ejemplos: <http://thehackerway.com/2011/05/23/exploando-vulnerabilidades-xss-en-aplicaciones-web/>

Inyección de Sql

La inyección de código SQL es un ataque en el cual se inserta código malicioso en las cadenas que posteriormente se pasan a una instancia de SQL Server para su análisis y ejecución. Todos los procedimientos que generan instrucciones SQL deben revisarse en busca de vulnerabilidades de inyección de código, ya que SQL Server ejecutará todas las consultas recibidas que sean válidas desde el punto de vista sintáctico. Un atacante cualificado y con determinación puede manipular incluso los datos con parámetros.

La forma principal de inyección de código SQL consiste en la inserción directa de código en variables especificadas por el usuario que se concatenan con comandos SQL y se ejecutan. Existe un ataque menos directo que inyecta código dañino en cadenas que están destinadas a almacenarse en una tabla o como metadatos. Cuando las cadenas almacenadas se concatenan posteriormente en un comando SQL dinámico, se ejecuta el código dañino.

El proceso de inyección consiste en finalizar prematuramente una cadena de texto y anexar un nuevo comando. Como el comando insertado puede contener cadenas adicionales que se hayan anexo al mismo antes de su ejecución, el atacante pone fin a

la cadena inyectada con una marca de comentario "--". El texto situado a continuación se omite en tiempo de ejecución.

En el siguiente script se muestra una sencilla inyección de código SQL. El script crea una consulta SQL concatenando cadenas no modificables con una cadena especificada por el usuario:

```
var Shipcity;  
ShipCity = Request.form ("ShipCity");  
var sql = "select * from OrdersTable where ShipCity = '" + ShipCity + "'";
```

Se le pide al usuario que escriba el nombre de una ciudad. Si el usuario especifica Redmond, la consulta ensamblada por el script presenta un aspecto similar al siguiente:

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond'
```

Sin embargo, suponga que el usuario especificase lo siguiente:

```
Redmond'; drop table OrdersTable--
```

En este caso, la siguiente consulta la ensambla el script:

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond';drop table OrdersTable--'
```

El punto y coma (;) denota el final de una consulta y el principio de otra. El guión doble (--) indica que el resto de la línea actual es un comentario y debe omitirse. Si el código modificado es sintácticamente correcto, el servidor lo ejecutará. Cuando SQL Server procese esta instrucción, SQL Server seleccionará en primer lugar todos los registros de OrdersTable donde ShipCity sea Redmond. A continuación, SQL Server quitará OrdersTable.

Siempre y cuando el código SQL inyectado sea sintácticamente correcto, no será posible detectar alteraciones mediante programación. Por ello, debe validar todos los datos especificados por el usuario y revisar cuidadosamente el código que ejecute comandos SQL construidos en el servidor que utilice.

Prácticas recomendadas de codificación para Sql Server 2008 R2:

Valide siempre los datos especificados por el usuario mediante comprobaciones de tipo, longitud, formato e intervalo. A la hora de implementar medidas de precaución frente a la especificación de datos dañinos, tenga en cuenta la arquitectura y los escenarios de implementación de la aplicación. Recuerde que los programas diseñados para ejecutarse en un entorno seguro pueden copiarse en un entorno no seguro.

Las sugerencias que se muestran a continuación deben considerarse prácticas recomendadas:

- No haga suposiciones sobre el tamaño, tipo o contenido de los datos que recibirá la aplicación. Por ejemplo, debe hacer la siguiente evaluación:

 - Cómo se comportará la aplicación si un usuario (malicioso o no) especifica un archivo MPEG de 10 megabytes cuando la aplicación espera un código postal.

 - Cómo se comportará la aplicación si se incrusta una instrucción DROP TABLE en un campo de texto.

 - Compruebe el tamaño y el tipo de los datos especificados y aplique unos límites adecuados. Esto puede impedir que se produzcan saturaciones deliberadas del búfer.

 - Compruebe el contenido de las variables de cadena y acepte únicamente valores esperados. Rechace las especificaciones que contengan datos binarios, secuencias de escape y caracteres de comentario. Esto puede impedir la inyección de scripts y puede servir de protección frente a explotaciones de saturación del búfer.

- Cuando trabaje con documentos XML, valide todos los datos con respecto a su esquema a medida que se vayan indicando.

- No cree nunca instrucciones Transact-SQL directamente a partir de datos indicados por el usuario.

- Utilice procedimientos almacenados para validar los datos indicados por el usuario.

- En entornos de varios niveles, todos los datos deben validarse antes de que se admitan en la zona de confianza. Los datos que no superen el proceso de validación deben rechazarse, y debe devolverse un error al nivel anterior.

- Implemente varias capas de validación. Las precauciones que tome contra usuarios malintencionados ocasionales pueden resultar ineficaces contra piratas informáticos con determinación. Lo más recomendable es validar los datos especificados por el usuario en la interfaz de usuario y, después, en todos los puntos posteriores en que atraviesen un límite de confianza.

Por ejemplo, la validación de datos en una aplicación de cliente puede evitar la inyección de scripts. Sin embargo, si en el siguiente nivel se asume que ya se ha validado la entrada, cualquier usuario malintencionado que sea capaz de eludir un cliente puede disfrutar de un acceso sin restricciones a un sistema.

- No concatene nunca datos especificados por el usuario que no se hayan validado. La concatenación de cadenas es el punto de entrada principal de una inyección de scripts.

-No acepte las siguientes cadenas en campos a partir de los que puedan construirse nombres de archivo: AUX, CLOCK\$, COM1 a COM8, CON, CONFIG\$, LPT1 a LPT8, NUL y PRN.

-Si es posible, rechace los datos que contengan los siguientes caracteres.

Carácter de entrada	Significado en Transact-SQL
;	Delimitador de consultas.
'	Delimitador de cadenas de datos de caracteres.
--	Delimitador de comentarios.
/* ... */	Delimitadores de comentarios. El servidor no evalúa el texto incluido entre /* y */.
xp_	Se utiliza al principio del nombre de procedimientos almacenados extendidos de catálogo, como xp_cmdshell.

-Usar parámetros SQL con seguridad de tipos

-La colección Parameters de SQL Server proporciona comprobación de tipos y validación de longitud. Si utiliza la colección Parameters, la entrada se considerará un valor literal en lugar de código ejecutable. Otra de las ventajas del uso de la colección Parameters es que puede exigir comprobaciones de tipos y de longitud. Los valores que no estén comprendidos en el intervalo desencadenarán una excepción. En el siguiente fragmento de código se muestra cómo utilizar la colección Parameters:

```
SqlDataAdapter myCommand = new SqlDataAdapter("AuthorLogin", conn);
myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
SqlParameter parm = myCommand.SelectCommand.Parameters.Add("@au_id",
    SqlDbType.VarChar, 11);
parm.Value = Login.Text;
```

En este ejemplo, el parámetro @au_id se considera un valor literal en lugar de código ejecutable. Se comprueba el tipo y la longitud de este valor. Si el valor de @au_id no cumple las restricciones especificadas de tipo y longitud, se producirá una excepción.

Usar una entrada con parámetros con los procedimientos almacenados

Los procedimientos almacenados pueden ser susceptibles de una inyección de código SQL si utilizan una entrada sin filtrar. Por ejemplo, el código que se muestra a continuación es vulnerable:

```
SqlDataAdapter myCommand =
new SqlDataAdapter("LoginStoredProcure '" +
```

```
Login.Text + """, conn);
```

Si utiliza procedimientos almacenados, deberá utilizar parámetros como entrada para dichos procedimientos.
Usar la colección Parameters con SQL dinámico

- Si no puede utilizar procedimientos almacenados, puede seguir utilizando parámetros, tal y como se muestra en el siguiente ejemplo de código.

```
SqlDataAdapter myCommand = new SqlDataAdapter(
"SELECT au_lname, au_fname FROM Authors WHERE au_id = @au_id", conn);
SqlParameter parm = myCommand.SelectCommand.Parameters.Add("@au_id",
    SqlDbType.VarChar, 11);
Parm.Value = Login.Text;
```

-Filtrar la entrada

Filtrar la entrada también puede ser útil para protegerse frente a inyecciones de código SQL mediante la eliminación de los caracteres de escape. Sin embargo, debido al gran número de caracteres que pueden presentar problemas, no se trata de una defensa confiable. El siguiente ejemplo busca el delimitador de cadenas de caracteres.

```
private string SafeSqlLiteral(string inputSQL)
{
    return inputSQL.Replace("'", "''");
}
```

-Cláusulas LIKE

Tenga en cuenta que si utiliza una cláusula LIKE, los caracteres comodín seguirán necesitando utilizar caracteres de escape:

```
s = s.Replace("[", "[");
s = s.Replace("%", "[%");
s = s.Replace("_", "[_");
```

-Revisar el código para la inyección de código SQL

Debe revisar todos los códigos que llaman a EXECUTE, EXEC o sp_executesql. Puede utilizar consultas similares a las siguientes para ayudarle a identificar los procedimientos

que contienen estas instrucciones. Esta consulta comprueba si hay 1, 2, 3 ó 4 espacios después de las palabras EXECUTE o EXEC.

```
SELECT object_Name(id) FROM syscomments
WHERE UPPER(text) LIKE '%EXECUTE (%)'
OR UPPER(text) LIKE '%EXECUTE (%)'
OR UPPER(text) LIKE '%EXECUTE (%)'
OR UPPER(text) LIKE '%EXECUTE (%)'
OR UPPER(text) LIKE '%EXEC (%)'
OR UPPER(text) LIKE '%EXEC (%)'
OR UPPER(text) LIKE '%EXEC (%)'
OR UPPER(text) LIKE '%EXEC (%)'
OR UPPER(text) LIKE '%SP_EXECUTESQL%'
```

-Incluir parámetros en QUOTENAME() y REPLACE()

En cada procedimiento almacenado seleccionado, compruebe que todas las variables que se utilizan en Transact-SQL dinámico se han administrado correctamente. Los datos procedentes de los parámetros de entrada del procedimiento almacenado o que se han leído desde una tabla deben estar incluidos en QUOTENAME() o REPLACE(). Recuerde que el valor de @variable que se pasa a QUOTENAME() es de tipo sysname, y tiene una longitud máxima de 128 caracteres.

@variable	Contenedor recomendado
Nombre de un asegurable	QUOTENAME(@variable)
Cadena de ≤ 128 caracteres	QUOTENAME(@variable, '"')
Cadena de > 128 caracteres	REPLACE(@variable, '"', '" "')

-Cuando se utiliza esta técnica, se pueden revisar las instrucciones SET como se indica a continuación:

--Before:

```
SET @temp = N'select * from authors where au_lname='''
+ @au_lname + N''''
```

--After:

```
SET @temp = N'select * from authors where au_lname='''
+ REPLACE(@au_lname, '"', '" "') + N''''
```

-Inyección habilitada mediante truncamiento de datos

Cualquier Transact-SQL dinámico que esté asignado a una variable se truncará si es mayor que el búfer asignado para dicha variable. Un atacante que pueda forzar el truncamiento de instrucciones pasando cadenas largas de forma inesperada a un procedimiento almacenado puede manipular el resultado. Por ejemplo, el procedimiento almacenado creado por el siguiente script es vulnerable a la inyección habilitada mediante truncamiento.

```
CREATE PROCEDURE sp_MySetPassword
```

```
@loginname sysname,
```

```
@old sysname,
```

```
@new sysname
```

```
AS
```

```
-- Declare variable.
```

```
-- Note that the buffer here is only 200 characters long.
```

```
DECLARE @command varchar(200)
```

```
-- Construct the dynamic Transact-SQL.
```

```
-- In the following statement, we need a total of 154 characters
```

```
-- to set the password of 'sa'.
```

```
-- 26 for UPDATE statement, 16 for WHERE clause, 4 for 'sa', and 2 for
```

```
-- quotation marks surrounded by QUOTENAME(@loginname):
```

```
-- 200 – 26 – 16 – 4 – 2 = 154.
```

```
-- But because @new is declared as a sysname, this variable can only hold
```

```
-- 128 characters.
```

```
-- We can overcome this by passing some single quotation marks in @new.
```

```
SET @command= 'update Users set password=' + QUOTENAME(@new, '') + ' where  
username=' + QUOTENAME(@loginname, '') + ' AND password = ' + QUOTENAME(@old,  
''')
```

```
-- Execute the command.
```

```
EXEC (@command)
```

```
GO
```

Al pasar 154 caracteres a un búfer de 128 caracteres, un atacante puede establecer una nueva contraseña para sa sin conocer la antigua.

```
EXEC          sp_MySetPassword          'sa',          'dummy',  
'1234567890123456789012345678901234567890123456789012345678901  
2345678901234567890123456789012'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Por este motivo, debe utilizar un búfer relativamente grande para una variable de comandos o para ejecutar directamente Transact-SQL dinámico dentro de la instrucción EXECUTE.

Truncamiento cuando se utilizan QUOTENAME(@variable, '') y REPLACE()

Las cadenas que devuelven QUOTENAME() y REPLACE() se truncarán sin avisar si exceden el espacio asignado. El procedimiento almacenado que se crea en el siguiente ejemplo muestra lo que puede suceder.

```
CREATE PROCEDURE sp_MySetPassword
```

@loginname sysname,

@old sysname,

@new sysname

AS

```
-- Declare variables.
```

DECLARE @login sysname

DECLARE @newpassword sysname

DECLARE @oldpassword sysname

```
DECLARE @command varchar(2000)
```

-- In the following statements, the data stored in temp variables

```
-- will be truncated because the buffer size of @login, @oldpassword,
```

-- and @newpassword is only 128 characters, but QUOTENAME() can return

-- up to 258 characters.

```
SET @login = QUOTENAME(@loginname, '')
```

```
SET @oldpassword = QUOTENAME(@old, '')
```

```
SET @newpassword = QUOTENAME(@new, '')
```

```
-- Construct the dynamic Transact-SQL.
```

-- If @new contains 128 characters, then @newpassword will be '123... n

-- where n is the 127th character.

```
-- Because the string returned by QUOTENAME() will be truncated,
```

-- it can be made to look like the following statement:

```
-- UPDATE Users SET password ='1234. . .[127] WHERE username=' -- other stuff here
```

```
SET @command = 'UPDATE Users set password = ' + @newpassword
+ ' where username = ' + @login + ' AND password = ' + @oldpassword;
-- Execute the command.
EXEC (@command)
```

GO

Por ello, la siguiente instrucción establecerá las contraseñas de todos los usuarios en el valor que se pasó en el código anterior.

```
EXEC          sp_MyProc          '--',          'dummy',
'1234567890123456789012345678901234567890123456789012345678901
234567890123456789012345678901234567890123456789012345678'
```

Puede forzar el truncamiento de cadenas al exceder el espacio en el búfer asignado cuando utiliza REPLACE(). El procedimiento almacenado que se crea en el siguiente ejemplo muestra lo que puede suceder.

```
CREATE PROCEDURE sp_MySetPassword
```

@loginname sysname,

@old sysname,

@new sysname

AS

```
-- Declare variables.
```

DECLARE @login sysname

DECLARE @newpassword sysname

```
DECLARE @oldpassword sysname
```

DECLARE @command varchar(2000)

-- In the following statements, data will be truncated because

```
-- the buffers allocated for @login, @oldpassword and @newpassword
```

-- can hold only 128 characters, but QUOTENAME() can return

-- up to 258 characters.

```
SET @login = REPLACE(@loginname, '"', '"\"')

```

```
SET @oldpassword = REPLACE(@old, '"', '""')
```

```
SET @newpassword = REPLACE(@new, '', '''')
```

```
-- Construct the dynamic Transact-SQL.
```

```
-- If @new contains 128 characters, @newpassword will be '123...n
```

```
-- where n is the 127th character.
-- Because the string returned by QUOTENAME() will be truncated, it
-- can be made to look like the following statement:
-- UPDATE Users SET password='1234...[127] WHERE username=' -- other stuff here
SET @command= 'update Users set password = '' + @newpassword + '' where
username='''
+ @login + '' AND password = '' + @oldpassword + ''';
-- Execute the command.
EXEC (@command)
GO
```

Como en el caso de QUOTENAME(), se puede evitar el truncamiento de cadenas por REPLACE() al declarar variables temporales que sean lo suficientemente grandes para todos los casos. Si es posible, debe llamar a QUOTENAME() o REPLACE() directamente dentro de Transact-SQL dinámico. De lo contrario, puede calcular el tamaño de búfer necesario del siguiente modo. Para @outbuffer = QUOTENAME(@input), el tamaño de @outbuffer debe ser $2 * (\text{len}(\text{@input}) + 1)$. Cuando utilice REPLACE() y las comillas dobles, como en el ejemplo anterior, bastará con un búfer de $2 * \text{len}(\text{@input})$.

En el siguiente cálculo se cubren todos los casos:

```
While len(@find_string) > 0, required buffer size =
round(len(@input)/len(@find_string),0) * len(@new_string)
+ (len(@input) % len(@find_string))
```

Truncamiento cuando se utiliza QUOTENAME(@variable, 'J')

El truncamiento se puede producir cuando el nombre de un asegurable de SQL Server se pasa a las instrucciones que utilizan el formato QUOTENAME(@variable, 'J'). Esto se muestra en el ejemplo siguiente.

```
CREATE PROCEDURE sp_MyProc
@schemaname sysname,
@tablename sysname,
AS
-- Declare a variable as sysname. The variable will be 128 characters.
-- But @objectname actually must allow for 2*258+1 characters.
DECLARE @objectname sysname
SET @objectname = QUOTENAME(@schemaname)+'.'+ QUOTENAME(@tablename)
```


-- Do some operations.

GO

Cuando se concatenan valores de tipo sysname, se deben utilizar variables temporales que sean lo suficientemente grandes como para comprender el máximo de 128 caracteres por valor. Si es posible, se debe llamar a QUOTENAME() directamente dentro de Transact-SQL dinámico. De lo contrario, puede calcular el tamaño de búfer necesario.

Este es un resumen de los principales incidentes de seguridad de los últimos años.

<http://www.hacktimes.com/principales-incidentes-seguridad/>

Análisis de Vulnerabilidades: Hay distintas maneras de analizar la vulnerabilidad de un servidor web y/o de una aplicación web.

- Contratación de servicios.
- Uso de software diseñado a tal efecto.

Software para análisis de vulnerabilidades:

Acunetix: <http://www.acunetix.com/ordering/>

Partner: Oficina en BUENOS AIRES

ComPlus S.A.

Leandro N Alem 530 - Piso 9

Buenos Aires 1001

Argentina

Tel: 54 11 5275-8800

jmesch@complus-arg.com.ar

<http://www.complus-arg.com.ar>

Security Guardian: <http://www.security-guardian.com/>

Nexpose: <http://www.rapid7.com/contact/>

Comparativa de funcionalidades:

http://www.hacktimes.com/scanners_de_seguridad_web_2012/

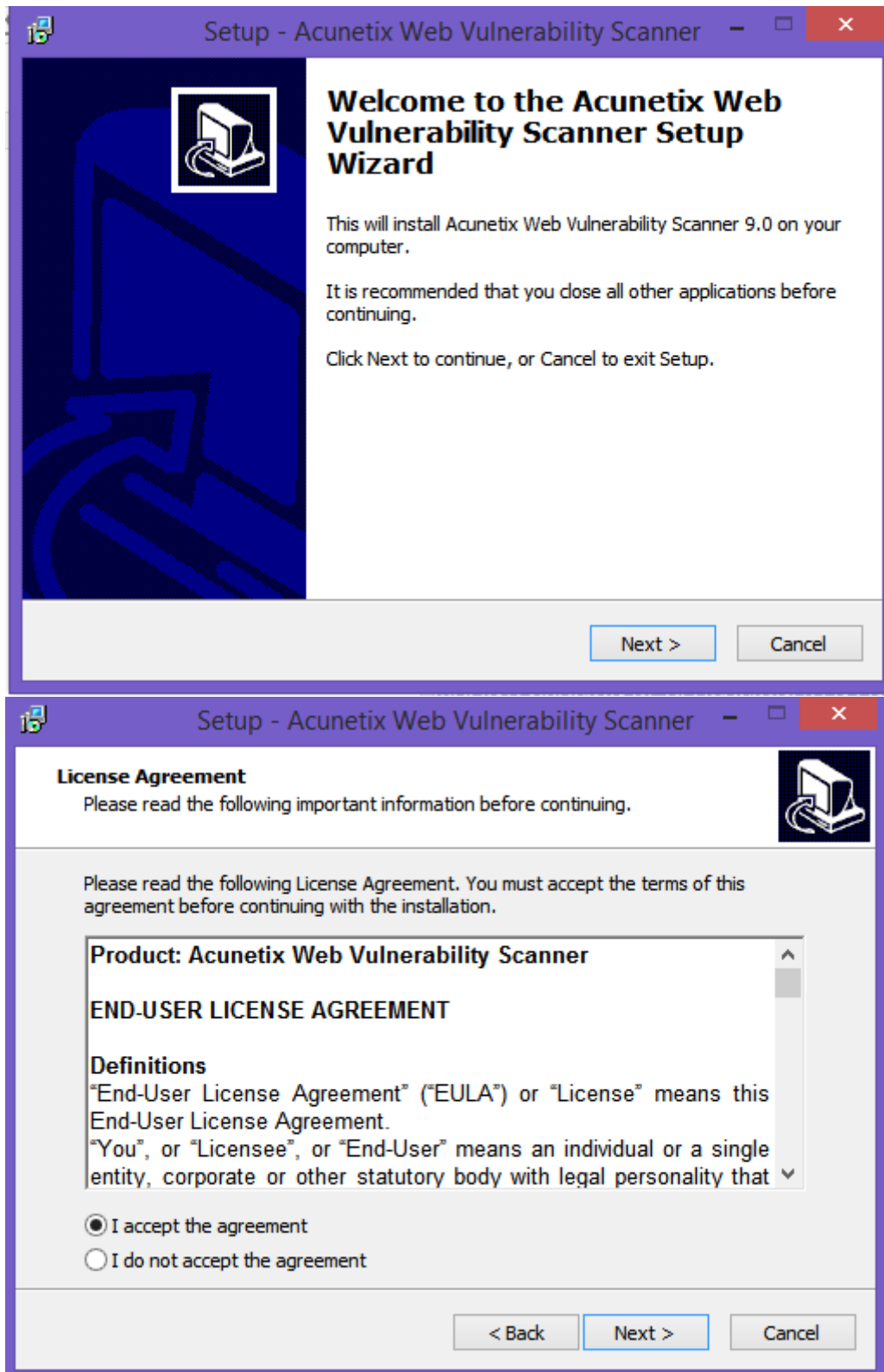
Comparativa de precios:

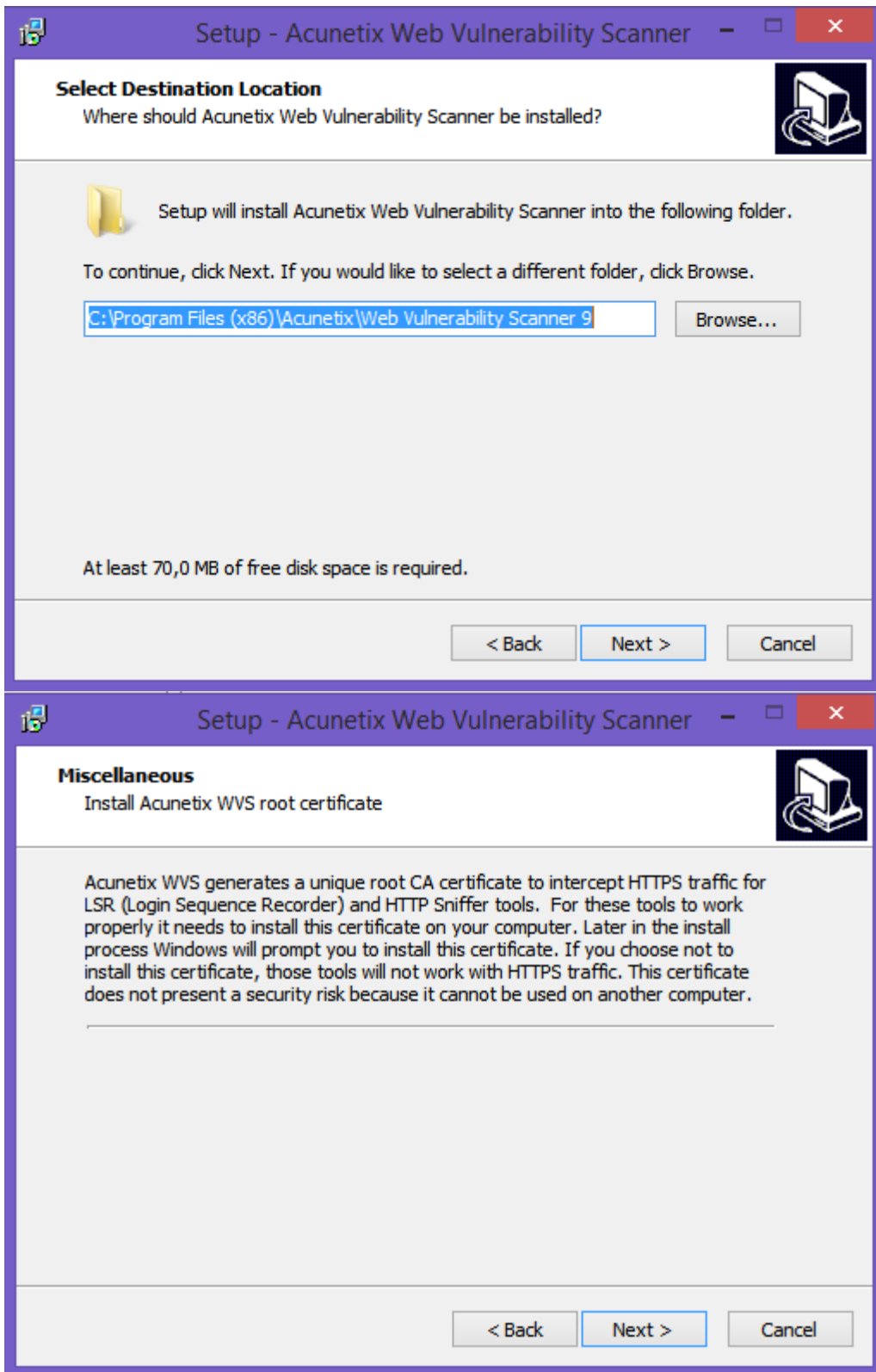
<http://www.alliancetechnpartners.com/acunetixcomparisontable.pdf>

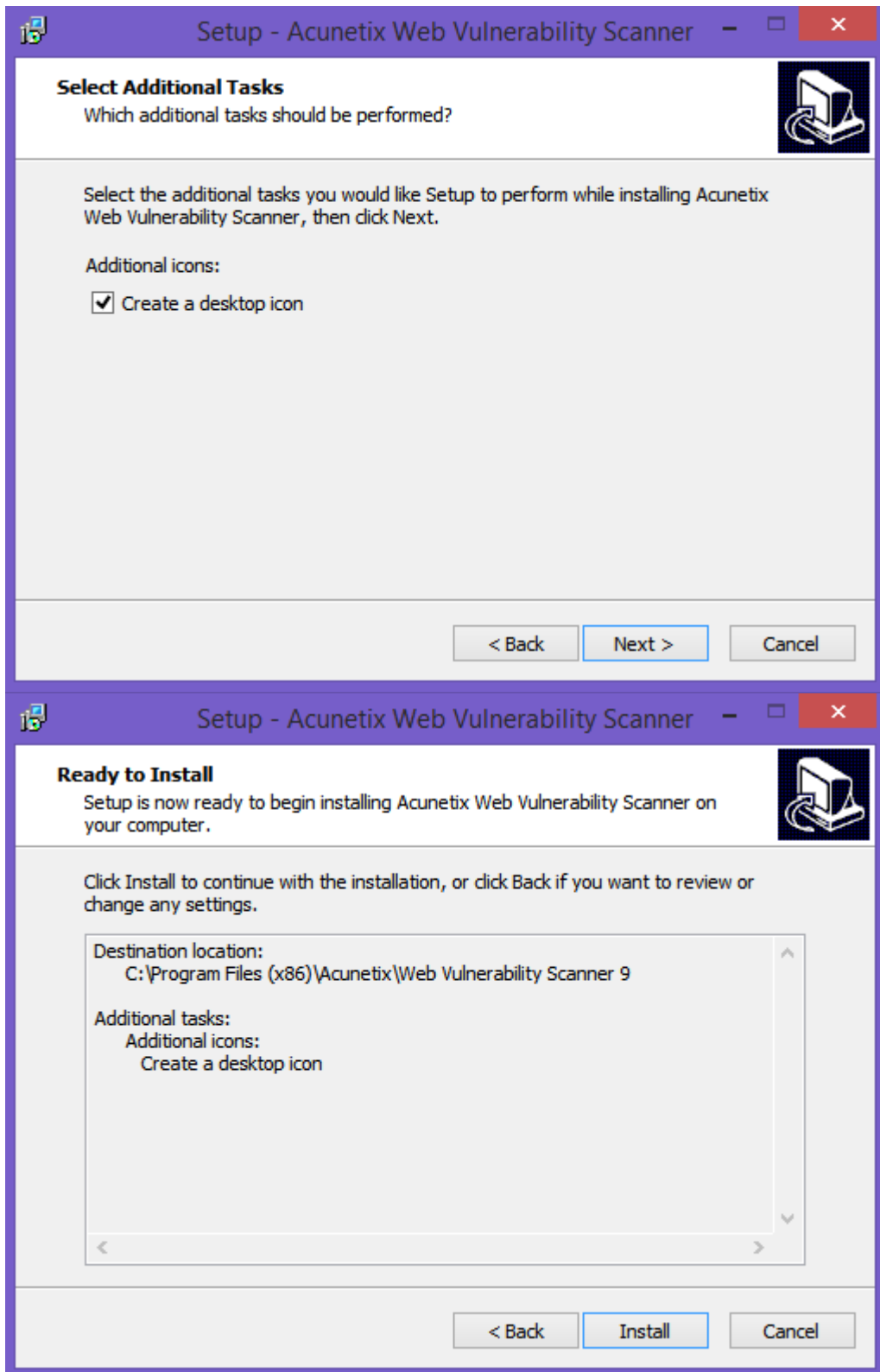
En el espacio Drive a continuación se encuentra el instalador del Software Acunetix:

<https://drive.google.com/open?id=0B93qrAESIWVvfjNaMHIDU2ZWZE50YjlxRjVaMGJkWU5CSFc1cUV4X3VhTDV0RVd0YnRKMIU>

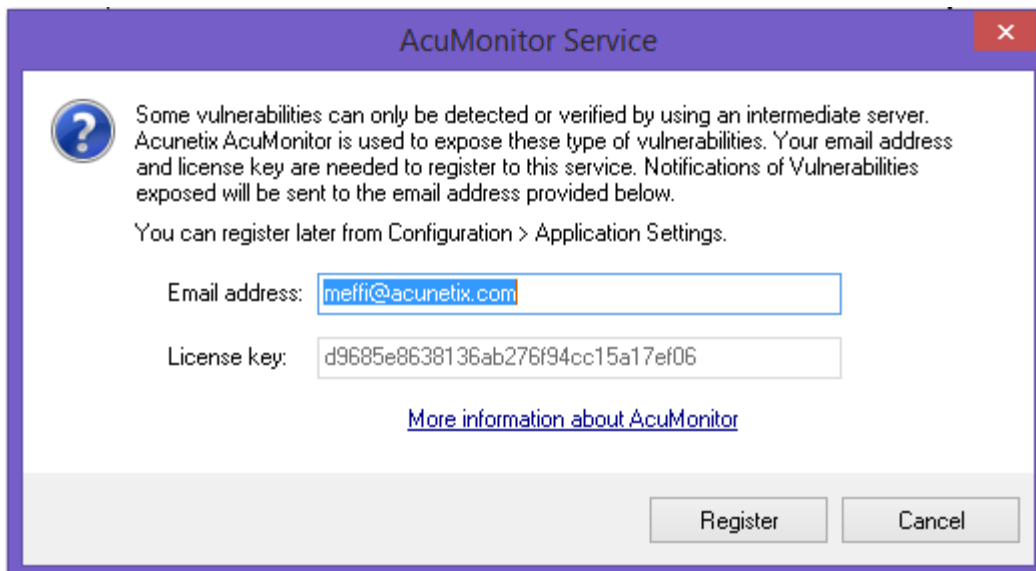
La instalación ofrece un conjunto de pantallas simples:







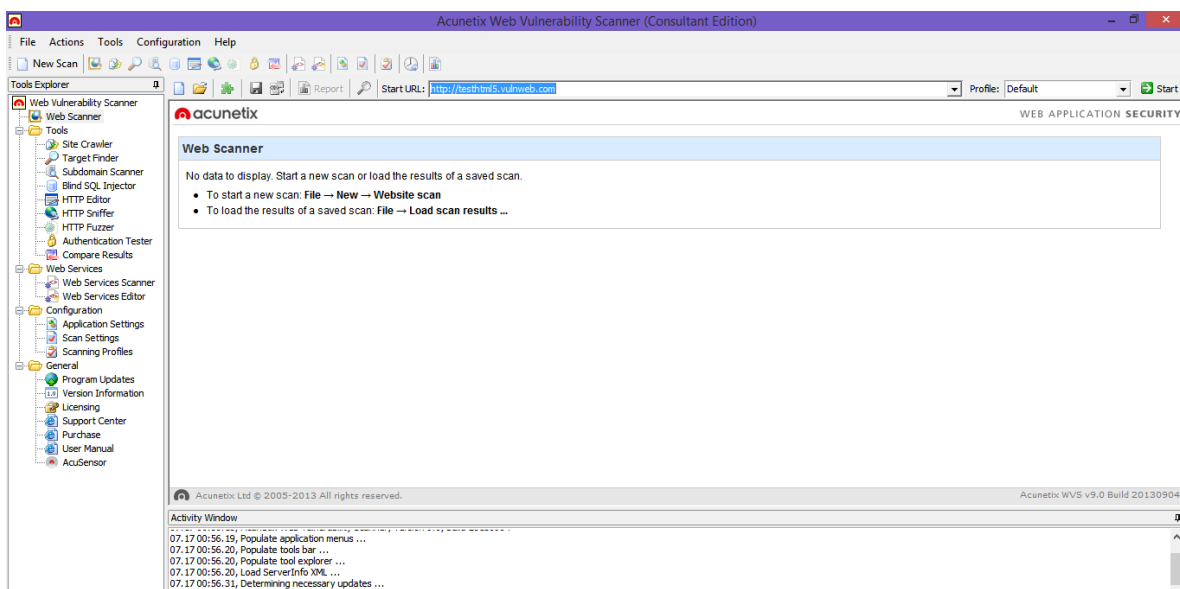
Al ejecutarlo ofrece el siguiente mensaje, incluso es posible actualizar las vulnerabilidades que detecta.



Guía básica de uso de Acunetix

Nuevo Scan:

Para iniciar un análisis de servidor o de aplicación presionar el botón "New Scan".



Escribir la URL del sitio que se desea analizar:

Scan Wizard

Scan Type

Select whether you want to scan a single website or analyze the results of a previous crawl.

Scan type


Here you can scan a single website. In case you want to scan a single web application and not the whole site you can enter the full path below. The application supports HTTP and HTTPS websites.

☒ Scan single website

Website URL:

If you saved the site structure using the site crawler tool you can use the saved results here. The scan will load this data from the file instead of crawling the site again.

☐ Scan using saved crawling results

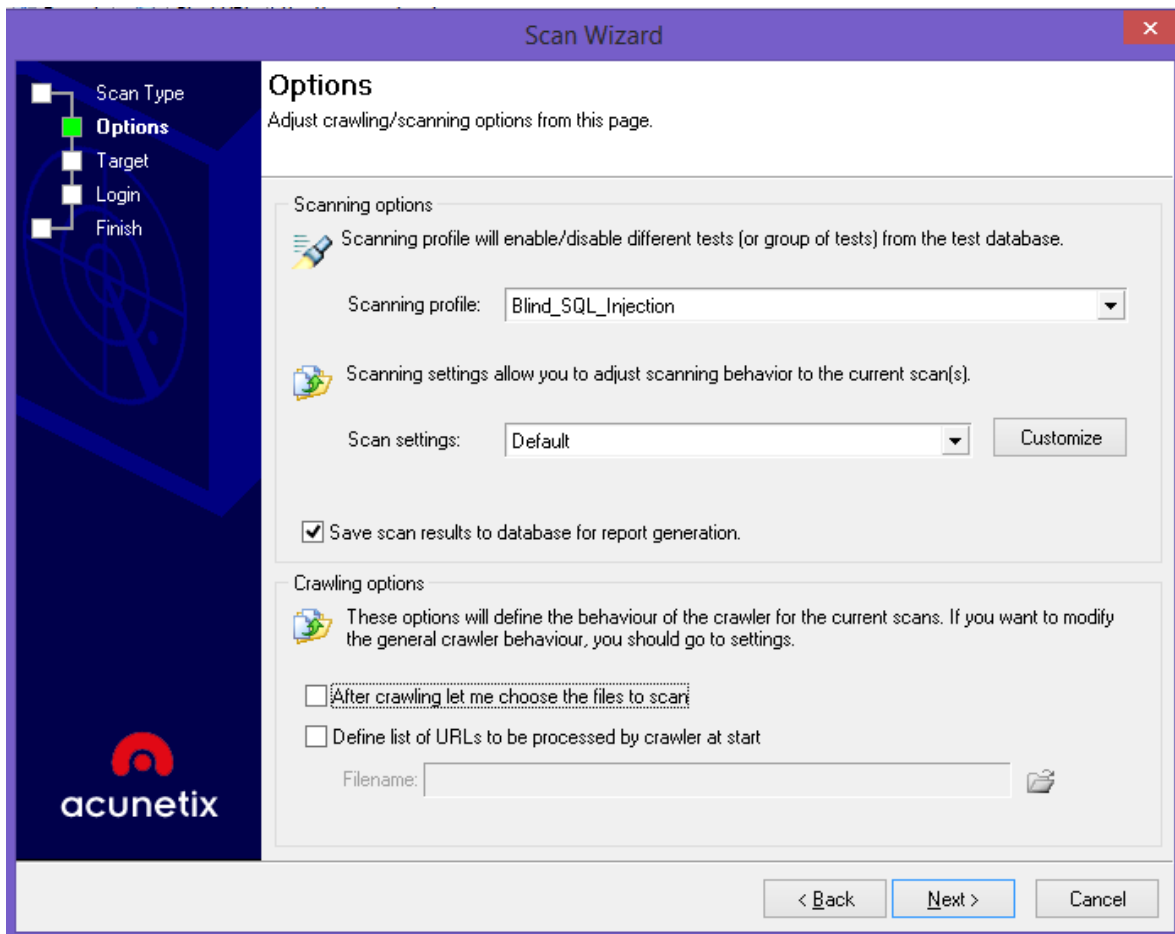
Filename: 

If you want to scan a list of websites, use the Acunetix Scheduler. You can access the scheduler interface by clicking the link below.

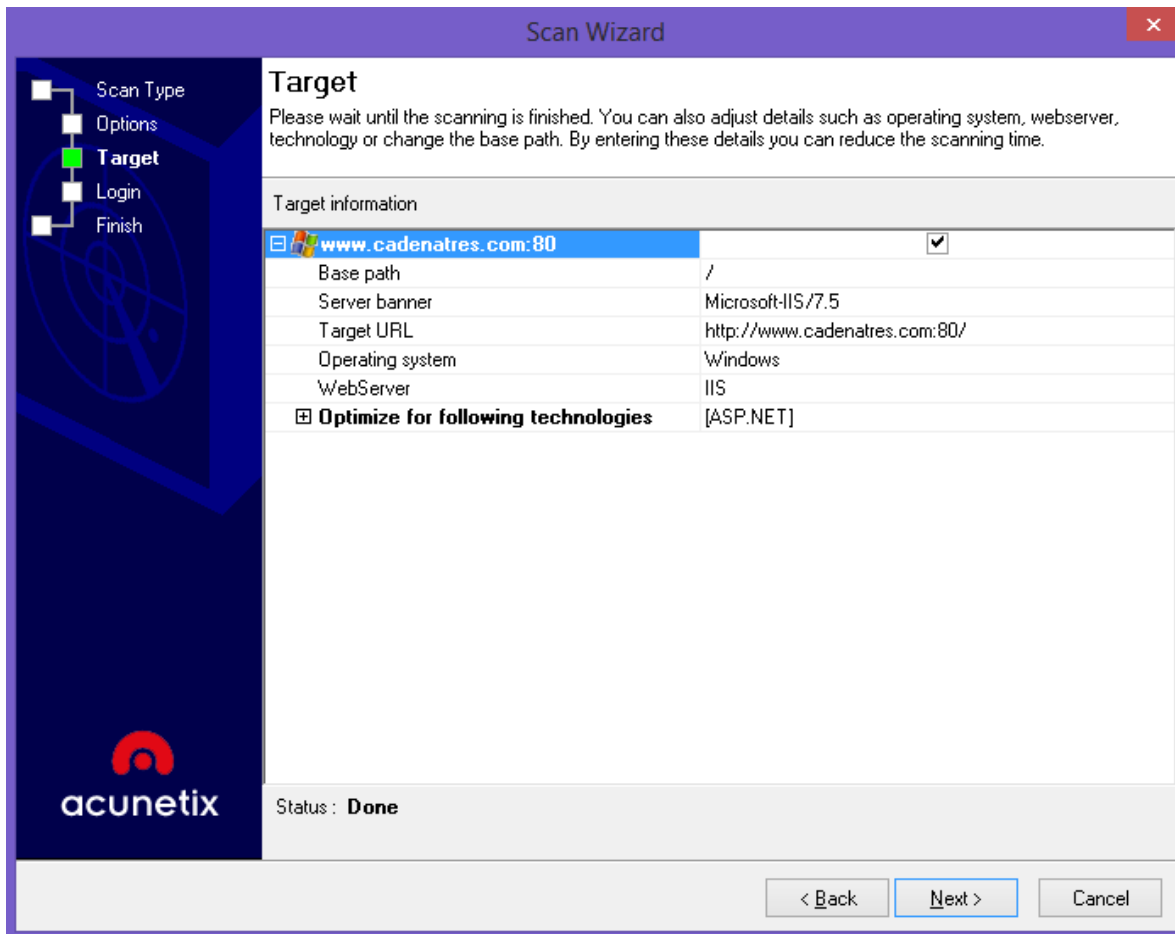
<http://localhost:8182/>

< Back Next > Cancel

Seleccionar el tipo de análisis que se desea llevar a cabo (Profile)



Tildar la opción Save Scan permite guardar los datos del análisis efectuado a fin de generar un reporte.



En el caso de conocer una secuencia de login (usuario y contraseña) puede grabarse y seleccionarse. También se puede seleccionar la modalidad de intervención manual (ingresar usuario y password en el momento en que se requiera). En el ejemplo estas observaciones no aplican.

Las vulnerabilidades encontradas se pueden guardar presionando el botón grabar. Las mismas se clasifican:

Verde – De carácter informativo.

Azul – Riesgo bajo.

Naranja – Riesgo medio.

Rojo – Riesgo alto.

De acuerdo a la exigencia de la empresa que lo requiere, en general, no se permiten los riesgos alto y medio.

La siguiente pantalla muestra el avance de la corrida del test.

Acunetix Web Vulnerability Scanner (Consultant Edition)

Tools Explorer

Web Vulnerability Scanner

Tools

Site Crawler

Target Finder

Subdomain Scanner

Blind SQL Injector

HTTP Editor

HTTP Sniffer

HTTP Fuzzer

Authentication Tester

Compare Results

Web Services

Web Services Scanner

Web Services Editor

Configuration

Application Settings

Scan Settings

Scanning Profiles

General

Program Updates

Version Information

Licensing

Support Center

Purchase

User Manual

AcuSensor

Scan Results

File upload (1)

Broken links (21)

Password type input with auto-complete enabled (4)

Knowledge Base

Site Structure

/

admin

clasificados

contenido

estilos

futbol

imagenes

imagenes

js

rss

1005fmcordoba.asp

923radopopular.asp

afscs-resolucion173-10.asp

audio.asp

audios.asp

avisos.asp

buscador.asp

cine.asp

clima.asp

comentar_ok_mundial2010.asp

Status

Moved Permanently

Moved Permanently

Moved Permanently

Moved Permanently

Moved Permanently

Moved Permanently

Moved Permanently

Moved Permanently

Moved Temporarily

Moved Temporarily

OK

Internal Server Error

Moved Permanently

Moved Permanently

Moved Permanently

Moved Permanently

Alerts summary

32 alerts

Acunetix Threat Level

Level 2: Medium

One or more medium-severity type vulnerabilities have been by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems.

Total alerts found

32

0 High

6 Medium

1 Low

25 Informational

Target information

http://www.cadenatres.com:80/

Statistics

1148 requests

Progress

Para cada vulnerabilidad hallada ofrece sugerencias de corrección.

Acunetix Web Vulnerability Scanner (Consultant Edition)

Tools Explorer

Web Vulnerability Scanner

Tools

Site Crawler

Target Finder

Subdomain Scanner

Blind SQL Injector

HTTP Editor

HTTP Sniffer

HTTP Fuzzer

Authentication Tester

Compare Results

Web Services

Web Services Scanner

Web Services Editor

Configuration

Application Settings

Scan Settings

Scanning Profiles

General

Program Updates

Version Information

Licensing

Support Center

Purchase

User Manual

AcuSensor

Scan Results

Scan Thread 1 (http://www.cadenatres.com:80/)

Web Alerts (32)

File upload (1)

Broken links (21)

Password type input with auto-complete enabled (4)

Knowledge Base

Site Structure

/

admin

clasificados

contenido

estilos

futbol

imagenes

imagenes

js

rss

1005fmcordoba.asp

923radopopular.asp

afscs-resolucion173-10.asp

audio.asp

audios.asp

avisos.asp

buscador.asp

Status

Moved Permanently

Forbidden

Forbidden

Forbidden

Moved Permanently

Forbidden

Forbidden

Moved Temporarily

Moved Temporarily

OK

Internal Server Error

Moved Permanently

Moved Permanently

Alerts summary

32 alerts

Acunetix Threat Level

Level 2: Medium

One or more medium-severity type vulnerabilities have been by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems.

Total alerts found

32

0 High

6 Medium

1 Low

25 Informational

Target information

http://www.cadenatres.com:80/

Statistics

1148 requests

Progress

File upload

Vulnerability description

This page allows visitors to upload files to the server. Various web applications allow users to upload files (such as pictures, images, sounds, ...). Uploaded files may pose a significant risk if not handled correctly. A remote attacker could send a multipart/form-data POST request with a specially-crafted filename or mime type and execute arbitrary code.

Affected items

- /comentar_ok_mundial2010.asp

The impact of this vulnerability

If the uploaded files are not safely checked an attacker may upload malicious files.

How to fix this vulnerability

Restrict file types accepted for upload; check the file extension and only allow certain files to be uploaded. Use a whitelist approach instead of a blacklist. Check for double extensions such as .php.png. Check for files without a filename like .htaccess (on ASP.NET, check for configuration files like web.config). Change the permissions on the upload folder so the files within it are not executable. If possible, rename the files that are uploaded.

Acunetix Web Vulnerability Scanner (Consultant Edition)

Tools Explorer

Web Vulnerability Scanner

Tools

Site Crawler

Target Finder

Subdomain Scanner

Blind SQL Injector

HTTP Editor

HTTP Sniffer

HTTP Fuzzer

Authentication Tester

Compare Results

Web Services

Web Services Scanner

Web Services Editor

Configuration

Application Settings

Scan Settings

Scanning Profiles

General

Program Updates

Version Information

Licensing

Support Center

Purchase

User Manual

AcuSensor

Scan Results

Scan Thread 1 (http://www.cadenatres.com:80/)

Web Alerts (34)

File upload (1)

Broken links (23)

Password type input with auto-complete enabled (4)

Knowledge Base

Site Structure

/

admin

clasificados

contenido

estilos

futbol

imagenes

imagenes

js

rss

1005fmcordoba.asp

923radopopular.asp

afscs-resolucion173-10.asp

audio.asp

audios.asp

avisos.asp

buscador.asp

Status

Moved Permanently

Forbidden

Forbidden

Forbidden

Moved Permanently

Forbidden

Forbidden

Moved Temporarily

Moved Temporarily

OK

Internal Server Error

Moved Permanently

Moved Permanently

Alerts summary

32 alerts

Acunetix Threat Level

Level 2: Medium

One or more medium-severity type vulnerabilities have been by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems.

Total alerts found

32

0 High

6 Medium

1 Low

25 Informational

Target information

http://www.cadenatres.com:80/

Statistics

1148 requests

Progress

Password type input with auto-complete enabled

Vulnerability description

When a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved. Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the cleartext password from the browser cache.

Affected items

- /contenido/2015/03/02/142140.asp (2330a13b3da95a23e5e6e68261af0088)
- /registro.asp

The impact of this vulnerability

Possible sensitive information disclosure

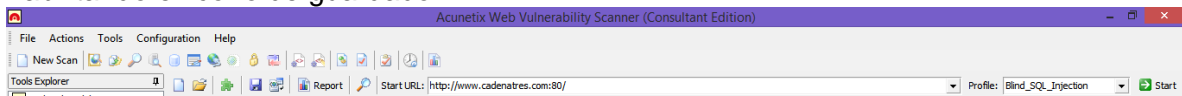
How to fix this vulnerability

The password auto-complete should be disabled in sensitive applications. To disable auto-complete, you may use a code similar to:

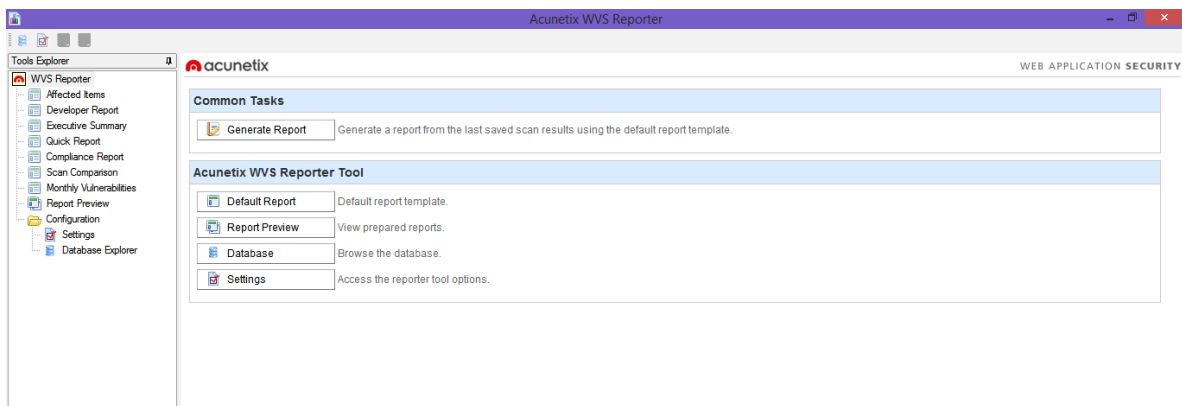
```
<INPUT TYPE="password" AUTOCOMPLETE="off">
```

Reportes:

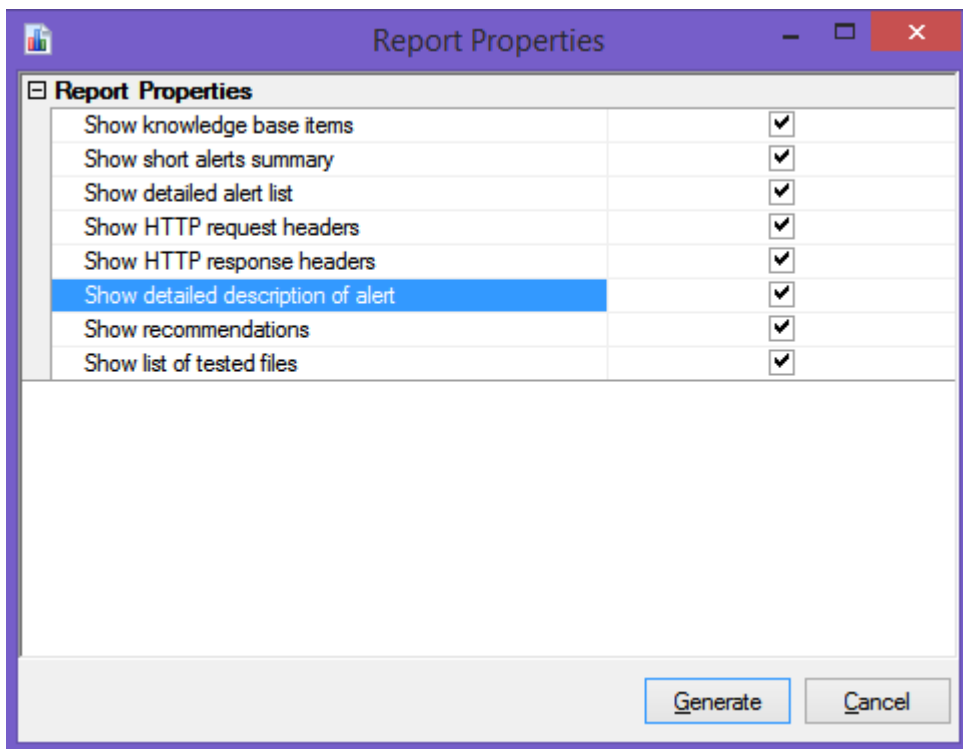
El botón Stop permite parar la corrida del test de vulnerabilidad y permite guardarlo habilitando el ícono de guardado.



La opción File - New – Report permite generar reportes a partir de un análisis generado. Es posible seleccionar distintos niveles de detalle del reporte de acuerdo al destinatario del informe.

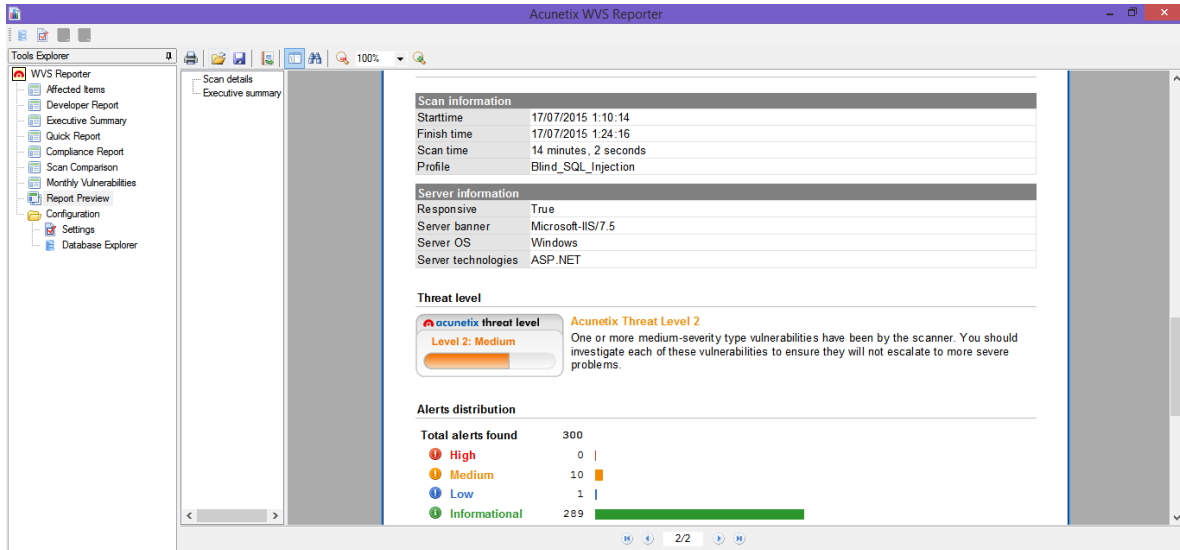


La opción Generate Reports ofrece mas ajustes.



Por defecto generará la versión Desarrollador con bastante detalle.

La opción Executive Summary es un informe bien breve tal como se visualiza a continuación:



Los reportes se guardan y luego de aplicar los cambios es factible compararlos.