Seguridad en aplicaciones web

Nombre Estudiante: Isidoro de la Cierva Rodriguez de Rivas

Programa: Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Nombre Consultor: Jordi Duch y Agusti Solanas

Centro: Universitat Oberta de Catalunya

Fecha entrega: 27/06/2014

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © AÑO TU-NOMBRE.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Seguridad en aplicaciones Web
Nombre del autor:	Isidoro de la Cierva Rodriguez de Rivas
Nombre del consultor:	Jordi Duch y Agusti Solanas
Fecha de entrega (mm/aaaa):	06/2014
Área del Trabajo Final:	Seguridad en servicios y aplicaciones
Titulación:	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Resumen del Trabajo (máximo 250 palabras):

El proyecto consiste en una plataforma web donde se puedan almacenar y lanzar escáneres contra servidores, para su control y análisis. El objetivo es poder tener de una manera centralizada y persistente la información de seguridad de los servidores de una organización.

La primera decisión que se tomó fue el analizador de seguridad a utilizar, que después de sopesar varias opciones se decidió que fuera Nessus, por tres razones: gratuito para uso personal, ampliamente utilizado y con interfaz de comunicación directo.

La segunda decisión, la plataforma tecnológica a utilizar, por la experiencia del desarrollador, se decidió utilizar tecnología Microsoft: ASP.NET, Microsoft SQL Server y Azure. La opción de publicarlo en la nube es opcional y se tomó para facilitar la demostración de los progresos alcanzados.

En el desarrollo del proyecto, se decidió que la pantalla principal mostrara primero los servidores presentes en el sistema, y a partir de ahí mostrar sus informes de escaneos y un detalle de las vulnerabilidades presentes en cada escaneo.

También se vio la necesidad de poder comparar entre diferentes escaneos para poder ver rápidamente las nuevas vulnerabilidades producidas en el tiempo.

Por supuesto tiene que ser posible el inicio de escaneos sobre los servidores, ya sean conocidos o nuevos en el sistema.

Por último se añadió funcionalidad de seguridad y configuración a la aplicación.

Una vez que esta la información en el sistema, se puede realizar de forma externa cualquier consulta o informe adicional sobre los datos almacenados.

Abstract (in English, 250 words or less):

The Project consists in a web platform where you can store and initiate scans against servers, for your control and analysis. The purpose is to be able to have centralized and persistently the organization's security information.

The first decision made was which security scanner to use, that after analyzing several options, we decided to use Nessus, for three reasons: It's free for personal use, it's widely used and it has a direct programming interface.

The second decision made was the technology platform to use, because our previous experience, we choose to use Microsoft technology: ASP.NET, Microsoft SQL Server y Azure. The option of publishing in the cloud was for making easier to other persons to test the application.

In the development of the Project, it was decided that the main window show first the servers present in the system, and from there show the report scans and a detail of every vulnerability present in each scan.

Also we saw the requirement to be able to compare different scan reports to see quickly the new vulnerabilities present in a system over the time.

Of course it has to be possible to initiate new server scans against old servers or new ones.

At last it was added security and configuration functionality to the application.

Once the information is in the system, it's possible to make externally any query or report against the stored data.

Palabras clave (entre 4 y 8):

Nessus, ASP .NET, XML/RPC, Security Scanner

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente existe una diversidad de escáneres de seguridad en el mercado, que permiten de una forma sencilla la realización de análisis de seguridad sobre aplicaciones web, pero no hay una plataforma centralizada que permita de una forma sencilla almacenar la información de los escaneos realizados para su posterior análisis y correlación.

Si en una organización existen por ejemplo 100 servidores, el llevar regularmente su seguridad y comprobar que vulnerabilidades nuevas aparecen o cuales son más urgentes de corregir, es una tarea ardua y difícil, ya que hay que manualmente comprobar cada informe y compararlo con informes anteriores para ver las novedades y el trabajo prioritario a realizar.

Actualmente si utilizamos Nessus, por ejemplo, se pueden ver los últimos escaneos ordenados por el tiempo, y el tipo de vulnerabilidades encontradas, con esos datos ya nos queda ir a cada informe y centrarnos en las más críticas para corregirlas primero.

Lo que se consigue con el proyecto, es tener una consola centralizada con todos los informes de escaneos de los servidores y poder ver fácilmente las nuevas vulnerabilidades, sin tener que revisarnos cada informe de cada servidor.

Al quedar almacenada la información en base datos, es susceptible de cualquier tipo de consulta o informe posterior, según las necesidades de cada organización.

1.2 Objetivos del Trabajo

Los objetivos del proyecto son:

- Desarrollar una plataforma web para la consulta, análisis y realización de escaneos de seguridad de aplicaciones web
- Se debe poder realizar nuevos escaneos sobre los servidores
- Se debe poder listar el historial de seguridad de un servidor
- Se debe poder comparar entre dos momentos en el tiempo de un servidor
- Se debe poder ver el detalle de las vulnerabilidades encontradas de un escaneo en concreto
- La aplicación debe tener seguridad para que sólo sea accesible por ciertos usuarios
- La aplicación debe poder ser configurable para poder especificar el servidor Nessus a utilizar y su información de acceso

1.3 Enfoque y método seguido

Los enfoques posibles pasaban por ver qué escáner de seguridad a utilizar y si era posible desarrollar un plugin en algún sistema para conseguir los objetivos.
Se decidió desarrollar una aplicación web independiente y no un plugin de un sistema externo para no depender completamente de un escáner en concreto, y poder migrar o utilizar varios escáneres para la recogida de información.
En cuanto al escáner a utilizar, tres opciones se contemplaron: Nmap, Metasploit y Nessus. Se decidió finalmente utilizar Nessus porque es un escáner ampliamente utilizado, es gratuito para uso personal y tiene una interfaz de programación que posibilita la interactuación con él de un modo programático.
Una vez elegido Nessus, quedaba por ver qué forma de comunicación utilizar para interactuar con el servidor, ya que hay varias, pero se decidió por utilizar la más directa y que se puede acceder mediante programación XML/RPC.
En cuanto a la arquitectura a utilizar, debido a que la experiencia del desarrollador es básicamente sobre entornos Microsoft, se decidió por ASP.NET y SQL Server.

1.4 Planificación del Trabajo

El trabajo ha sido realizado por un solo desarrollador, con la ayuda de dos consultores de la UOC para el análisis y la planificación del proyecto.
Las tareas a realizar son:

- Análisis de requisitos:
    - Estudio de la seguridad en aplicaciones web, utilizando la guía OWASP (10h)
    - Estudio de la problemática actual en la utilización de escáneres de seguridad (10h)
    - Estudio de arquitecturas existentes para la realización de aplicaciones web (5h)
- Desarrollo de la aplicación:
    - Aplicación web inicial ASP.NET (20h)
    - Comunicación via XML/RPC con Nessus (40h)
    - Almacenamiento de escaneos en base de datos (30h)
    - Visualización en la aplicación web de los resultados de los escaneos (40h)
    - Inicio de nuevos escaneos de seguridad (5h)
    - Comparación entre momentos distintos del historial de un servidor (5h)
    - Implementación de seguridad en la aplicación (10h)
    - Implementación de configuración en la aplicación (10h)
- Documentación de la aplicación:
    - Memoria de la aplicación (20h)
    - Presentación en Powerpoint (5h)

- Presentación en video (5h)

Lo que da un total de 215 horas, es decir alrededor de 27 dias de trabajo distribuidos a lo largo de 4 meses (Marzo, Abril, Mayo y Junio)

1.5 Breve sumario de productos obtenidos

El resultado obtenido es:

- Aplicación Web realizada en ASP.NET sobre una base de datos SQL Server
- Interfaz de comunicación con Nessus via XML/RPC
- Historial de escaneos de los servidores analizados
- Comparación entre las vulnerabilidades de un servidor a lo largo del tiempo
- Detalle de las vulnerabilidades de un servidor
- Seguridad integrada dentro de la aplicación
- Configuración dentro de la aplicación
- Publicación en la nube Azure para tests y demostraciones

1.6 Breve descripción de los otros capítulos de la memoria

En el resto de capítulos, se explicara la aplicación desde cuatro puntos de vista: EL interfaz web, la base de datos, el código de programación y la comunicación con Nessus, requiriendo un capítulo aparte debido a su importancia dentro del proyecto

2. Diseño de la aplicación Web

La aplicación web se ha desarrollado en ASP.NET 4.5, utilizando Web Forms con Model Binding. La experiencia anterior con estas herramientas había sido con ASP.NET 2.0, pero se decidió aprovechar las últimas tecnologías disponibles pese al mayor aprendizaje que esto implicaba. No se optó por utilizar MVC (que es la otra posibilidad que brinda ASP.NET), ya que ahí la curva de aprendizaje iba a ser más lenta, porque teníamos experiencia previa en la utilización Web Forms.
Como pantalla principal, teníamos que ver de forma clara los servidores presentes en el sistema y un resumen de los escaneos realizados sobre los mismos, con lo que queda un diseño como el de la Figura 1:

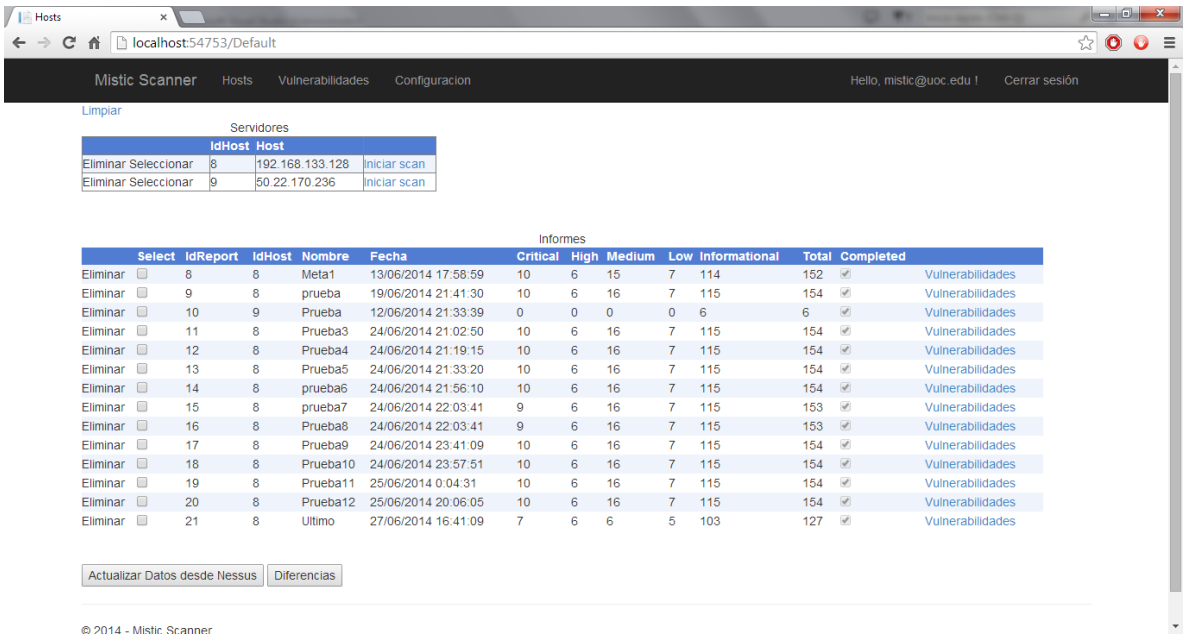

Figura 1

Como se puede ver, en un primer grid se muestran los servidores disponibles, y en un segundo grid todos los informes de escaneos realizados a los mismos.

Para actualizar los datos, pulsamos el botón “Actualizar Datos desde Nessus”, que desencadena una comunicación con el servidor de Nessus que actualiza todos los datos de escaneos realizados y los guarda en nuestra aplicación.

Si queremos ver solo el historial de un servidor en concreto, basta con seleccionar el mismo, para que su historial quede filtrado

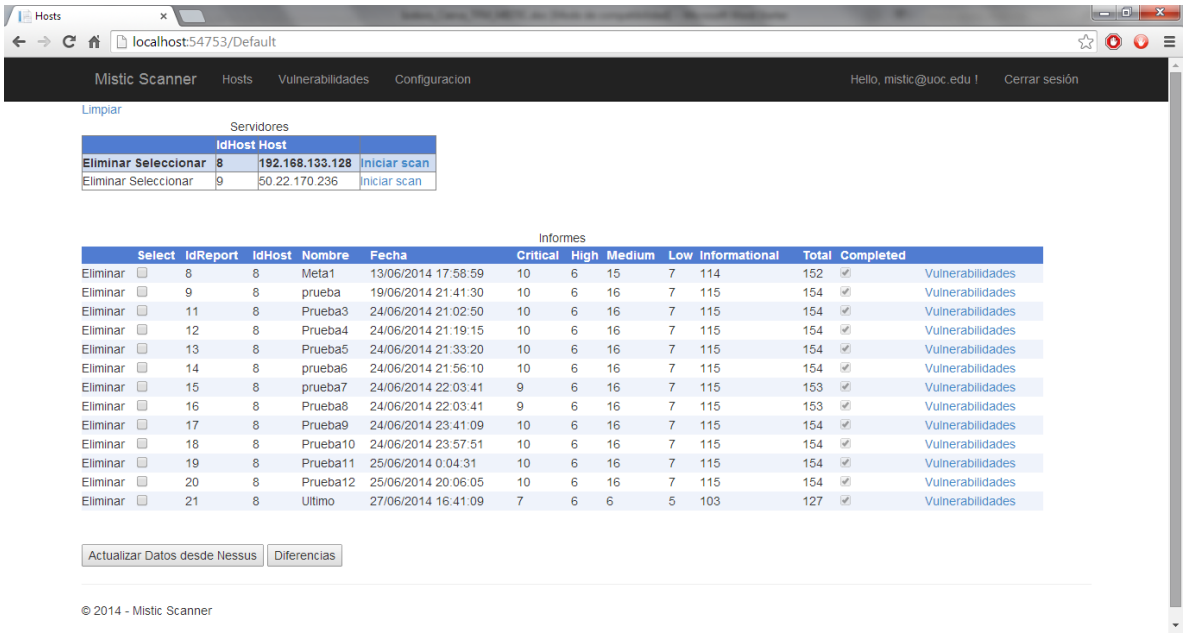

Figura 2

Si queremos volver a ver todos los informes disponibles, basta con pulsar el enlace “Limpiar” del grid de servidores.

En cada escaneo podemos ver en este resumen la fecha en que se realizó, un nombre descriptivo y el número de vulnerabilidades que contiene, divididas a su vez en categorías: Crítica, Alta, Media, Baja e Información. Así mismo se puede ver si el escaneo ha sido completado o todavía se está ejecutando.

Si queremos ver el detalle de un informe de escaneo, basta con pulsar el enlace “Vulnerabilidades” que nos muestra una nueva página con todas las vulnerabilidades que se encontraron en dicho escaneo

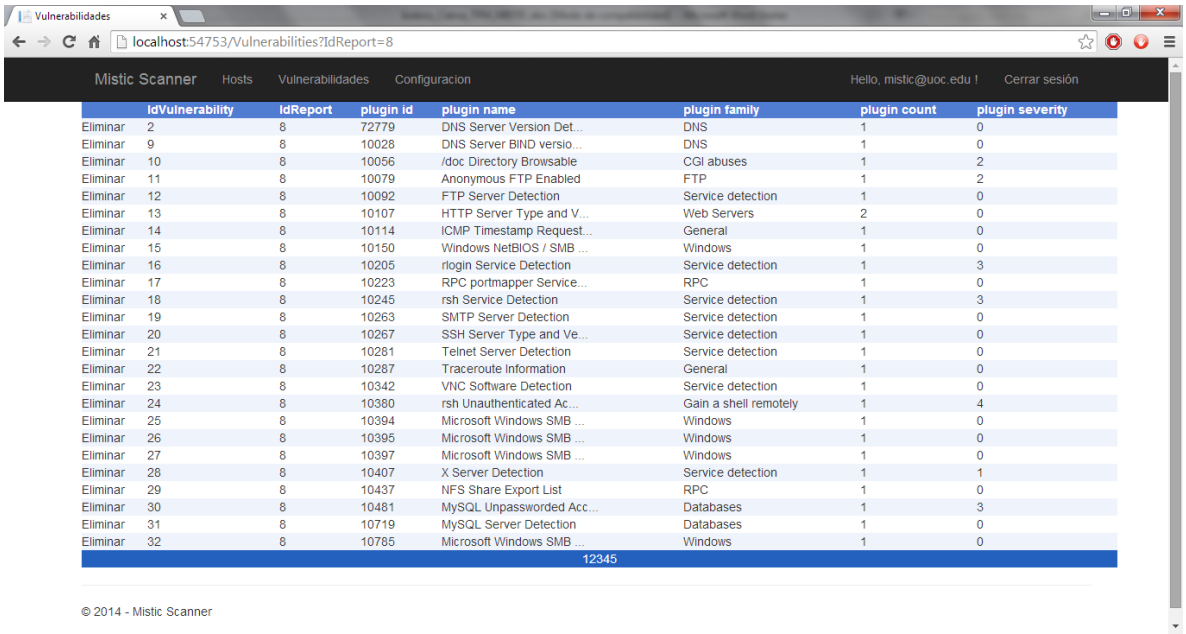

Figura 3

En la figura 3 podemos ver los detalles del escaneo, el identificador del plugin utilizado, su nombre, el grupo o familia al que pertenece, su numero y la criticidad del mismo
Si queremos realizar un nuevo escaneo, basta con pulsar el enlace “Iniciar scan” del servidor sobre el que lo queramos hacer, como podemos ver en la figura 4

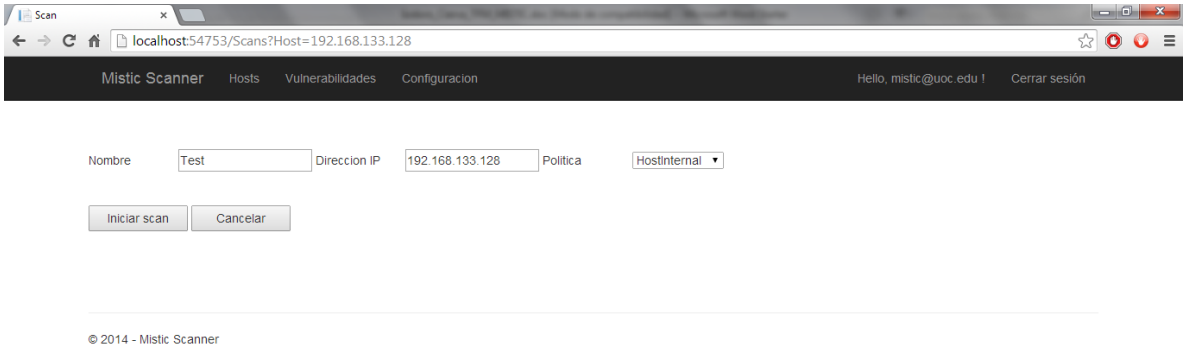

Figura 4

Aquí introducimos el nombre descriptivo del escaneo que se va a realizar, la dirección IP del servidor (podemos cambiarla para realizarla sobre servidores que no estén previamente en el sistema) y la política de Nessus que se va aplicar. Nessus utiliza estas políticas para diferenciar entre las distintas tipologías de escaneos, como interna, externa, descubrimiento de servidores, enumeración de puertos, etc.

Una vez iniciado, el servidor Nessus se encarga de ejecutarlo hasta que finalice, desde nuestra aplicación podemos ver el progreso del mismo y si se ha completado o no, actualizando los datos desde Nessus.

Si queremos comparar dos escaneos para ver sólo las nuevas vulnerabilidades descubiertas, basta con seleccionar dos informes de escaneos (por ejemplo los de los días 13 y 19 de junio), y pulsar el botón de “Diferencias”, con lo cual pasamos a ver una pantalla con los datos que nos interesan, como se puede ver en la figura 5:

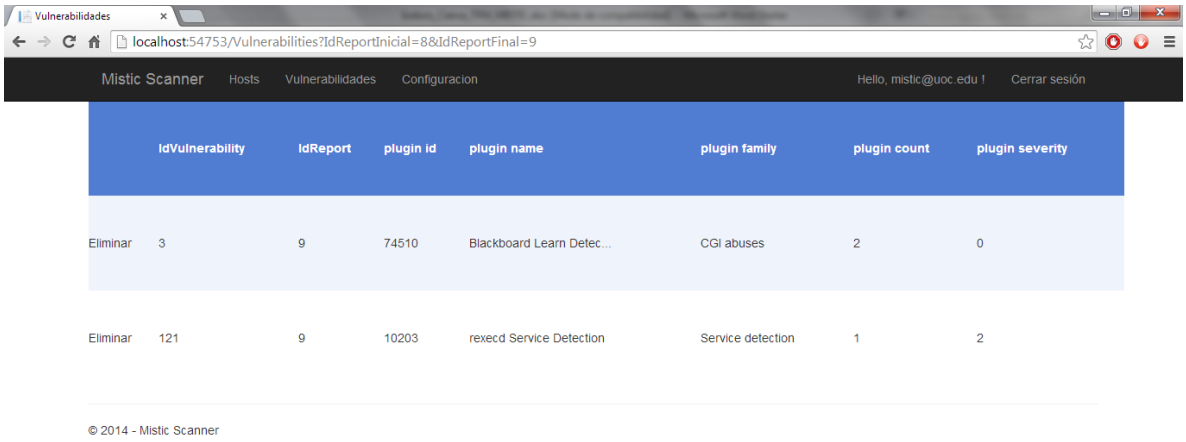

Figura 5

Como podemos ver, sólo se muestran las dos nuevas vulnerabilidades que se descubrieron en el escaneo del dia 19 con respecto al dia 13 de junio, que son una vulnerabilidad media y otra de tipo de información.

Si queremos configurar la aplicación, nos iremos a la opción de menú “Configuración” donde podremos especificar la dirección IP del servidor Nessus, y el usuario y contraseña con el que se accede al mismo, como se puede ver en la figura 6

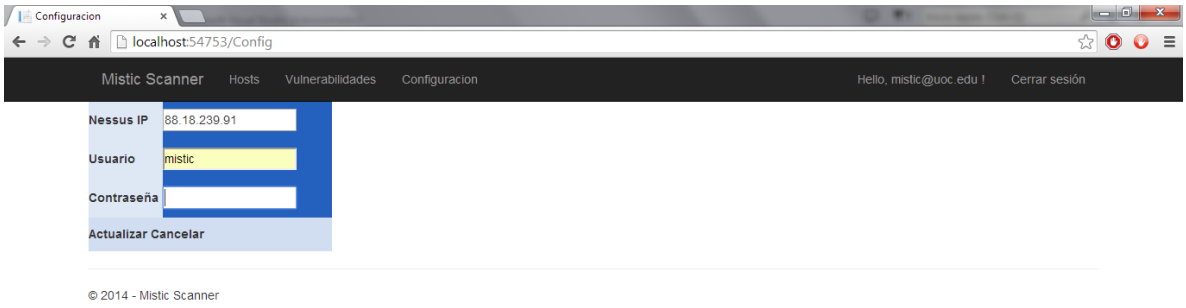

Figura 6

La seguridad de la aplicación esta controlada por las funcionalidades que nos aporta ASP.NET, proveniendonos automaticamente de funcionalidades pantalla

de login y de manejo de cuentas, en la figura 7 podemos ver la pantalla de entrada al sistema

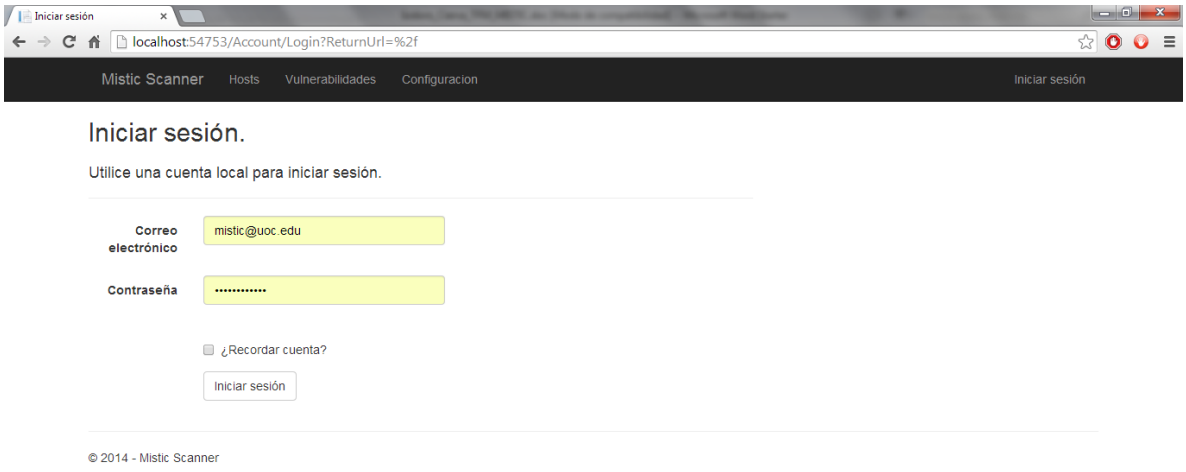

Figura 7

3. Diseño de la base de datos

Aunque realmente se ha diseñado la base de datos a través del código, ya que se ha utilizado una estrategia de diseño del modelo primero, que automáticamente se traspasa a la base de datos, vamos a explicar las principales entidades o modelos utilizadas en la aplicación.

Primero, vamos a ver un diagrama de la base de datos en la figura 8

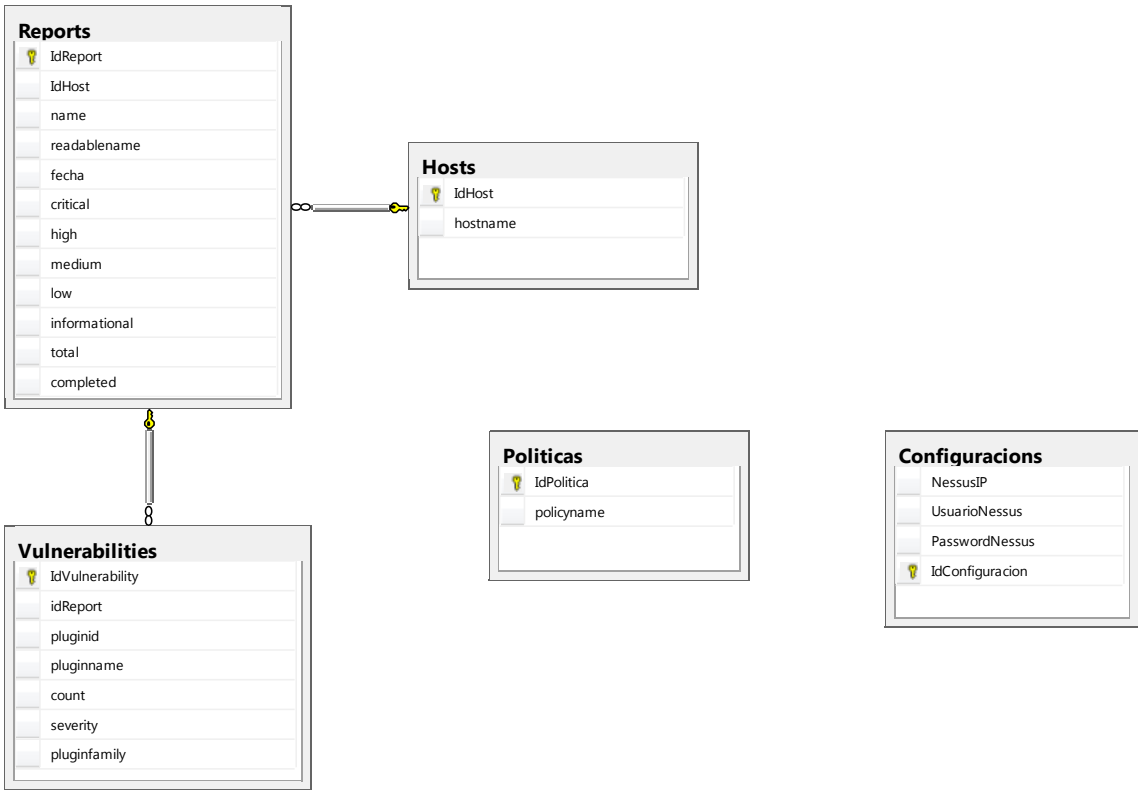

Figura 8

La primera tabla creada es la tabla Hosts, que contiene el nombre o dirección IP del servidor

La tabla Reports contiene los informes de los escaneos realizados a los servidores, y está relacionada con la tabla Hosts para saber a qué servidor pertenecen. En ella se detalla la fecha, criticidad de las vulnerabilidades encontradas, si está o no completado, y un campo muy importante, el campo “name” que contiene el UUID del informe almacenado en el servidor Nessus, por lo que esta información queda entrelazada de manera unívoca

La tabla Vulnerabilities contiene el detalle del escaneo del informe con el que se relaciona a través del identificador de informe “IdReport”. En ella se detallan el identificador de plugin que descubre la vulnerabilidad, su nombre, su grupo o familia, el número de veces que aparece y su criticidad.

La tabla Politicas se utiliza para almacenar las políticas existentes en Nessus, y poder tener esa información sin necesidad de comunicarse con él.

La tabla Configuracions almacena la configuración del sistema: la dirección IP del servidor Nessus, y el usuario y contraseña de acceso al mismo.

Para realizar la migración del Modelo almacenado en código a la base de datos, se ha utilizado la extensión NuGet Entity Framework de Visual Studio, que se encarga de tener sincronizada el modelo de código con la base de datos.

4. Diseño del código de la aplicación

La aplicación es un Web Forms con Model Binding en la capa de datos y apoyado sobre el Entity Framework, todo de Microsoft y siendo las últimas tecnologías disponibles para la realización de aplicaciones web. Como ya se ha dicho, se decidió no utilizar MVC debido a que teníamos ya una experiencia previa con Web Forms que aceleró su desarrollo.

Se podría haber utilizado tecnologías más antiguas y más conocidas por parte del desarrollador como ASP .NET 2.0, pero aunque el coste ha sido mayor, el beneficio ha sido también mucho mayor ya que se realizado una puesta al día en las tecnologías disponibles hoy en día por parte de Microsoft para la realización de aplicaciones web.

La base de datos elegida ha sido SQL Server, pero realmente poco trabajo se ha realizado sobre ella, ya que toda la lógica de los datos reside sobre las clases de modelo.

Para la comunicación con el servidor Nessus, se ha utilizado peticiones web estándar, con la dificultad previa de que Nessus utiliza un certificado no confiable, pero se solucionado satisfactoriamente. La comunicación se realiza mediante XML/RPC, y se ha utilizado las clases de .NET DataContractSerializer para la deserialización del XML de respuesta en las clases pertinentes, que se han creado previamente sobre los esquemas XML de respuesta de Nessus. En la figura 9 podemos ver las clases agrupadas en sus espacios de nombre y la relación de dependencia entre las mismas

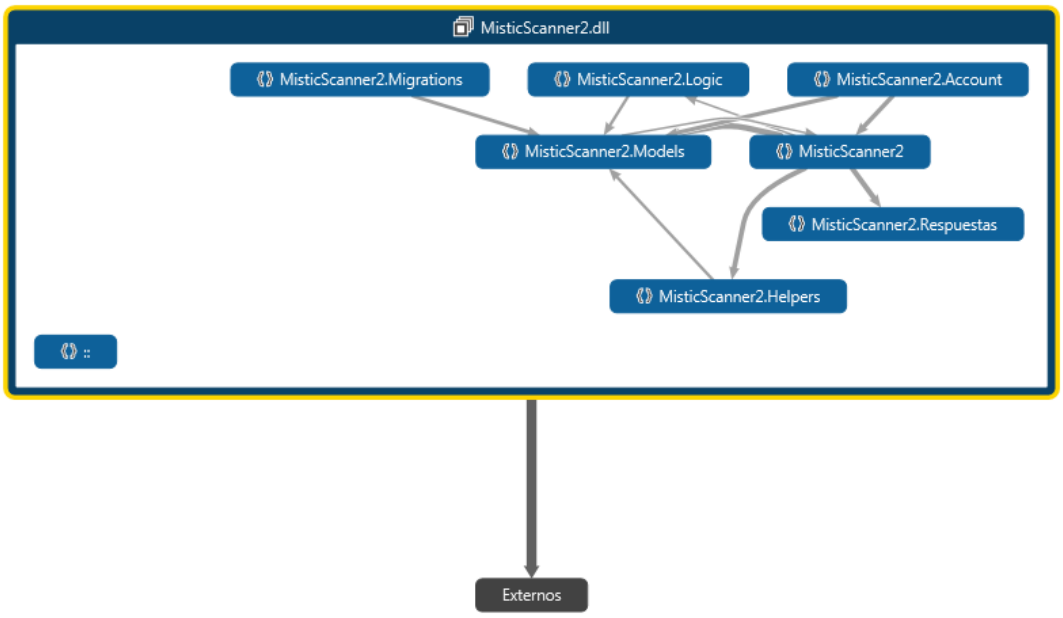

Figura 9

El código esta agrupado en varios espacios de nombres:

- MisticScanner2: Aquí están todos los formularios web utilizados, tanto los generados automáticamente como los creados manualmente por el desarrollador, destacando:
    - Config.aspx: Página de configuración de la aplicación
    - Default.aspx: Página principal de la aplicación, donde reside gran parte de su funcionalidad y desde donde se realiza la actualización de datos con el servidor Nessus. Es la que se encarga de coordinar con el resto de la aplicación: Base de datos, Servidor Nessus, conversión de XML, etc
    - Scans.aspx: Página desde la que se puede realizar un nuevo escaneo sobre un servidor seleccionado
    - Vulnerabilities.aspx: Página que contiene el detalle de las vulnerabilidades encontradas en un informe dado
    - Login.aspx: Página que se encarga de la seguridad del sistema, apoyada sobre las funcionalidades que nos da ASP.NET

  En la figura 10 podemos ver un diagrama de sus clases y dependencias

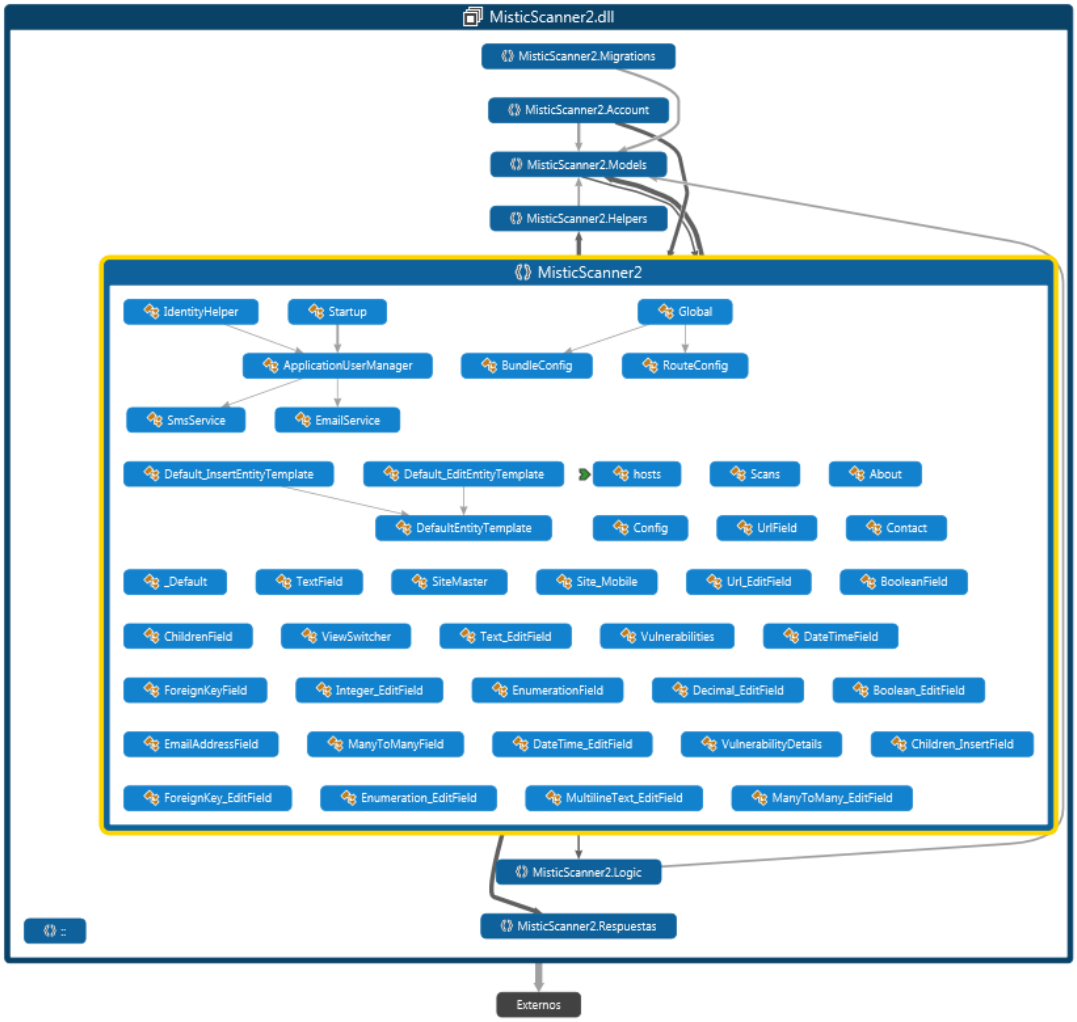

Figura 10

- Migrations: Contiene el código gestionado por el Entity Framework para tener sincronizados la base de datos y su modelo

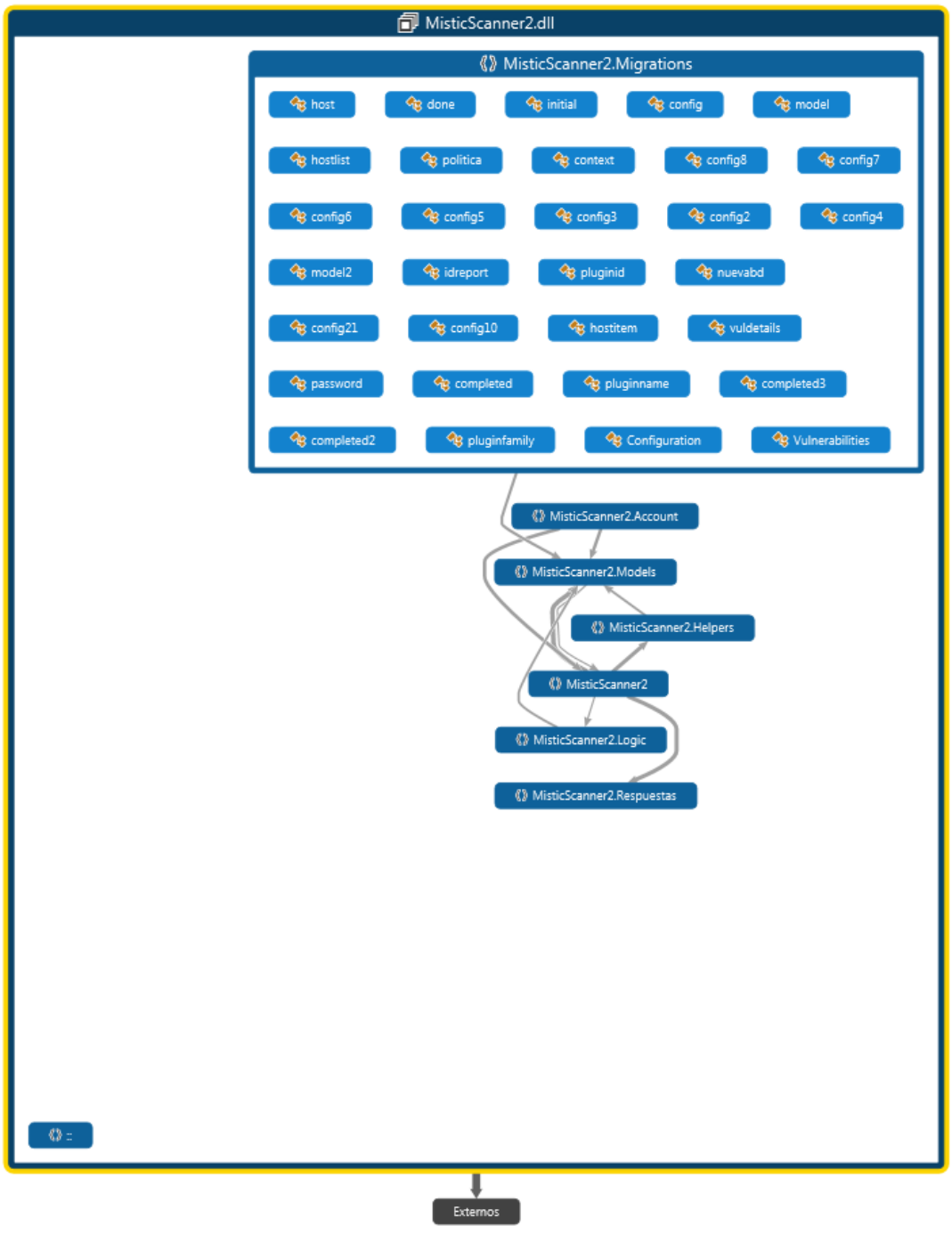

Figura 11

- Account: Contiene el código de ASP.NET Identity que gestiona la seguridad de la aplicación

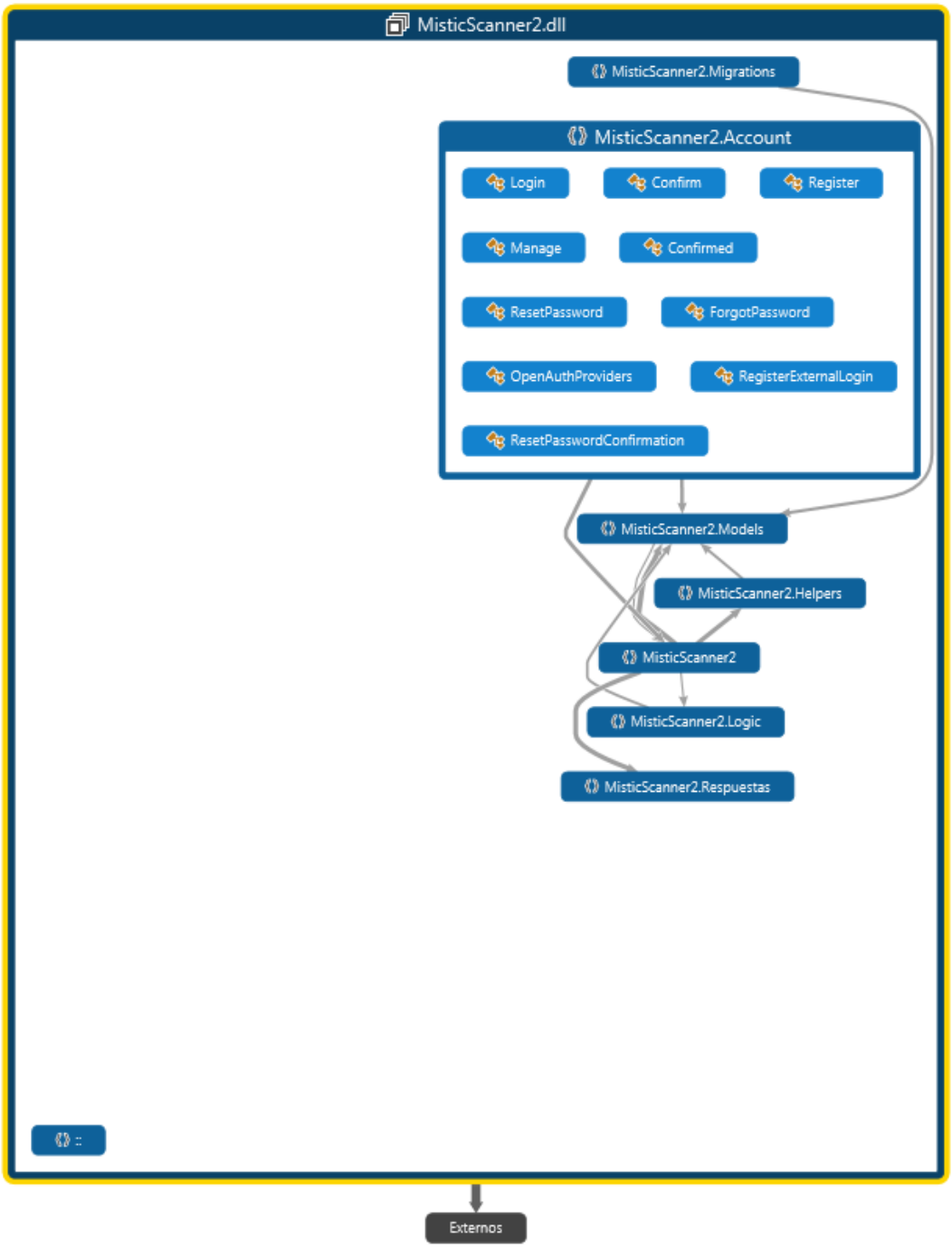

Figura 12

- Models: Contiene los modelos de la aplicación, que representan las entidades almacenadas en la base de datos y que se sincronizan automáticamente con la misma

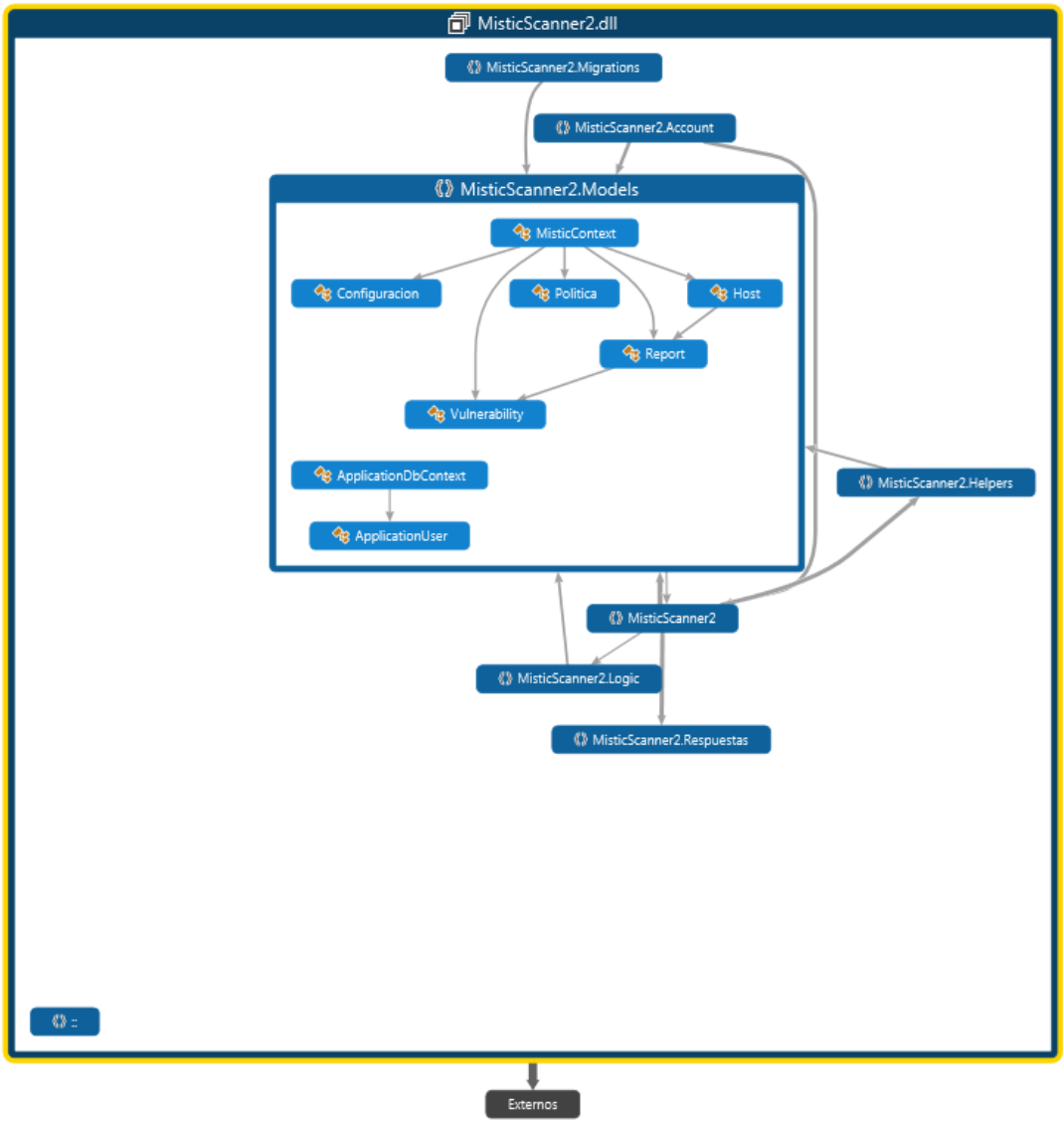

Figura 13

- Helpers: Contiene clases auxiliares, algunas de ellas tan importantes como la que se encarga de la comunicación con Nessus o la que deserializa las comunicaciones XML

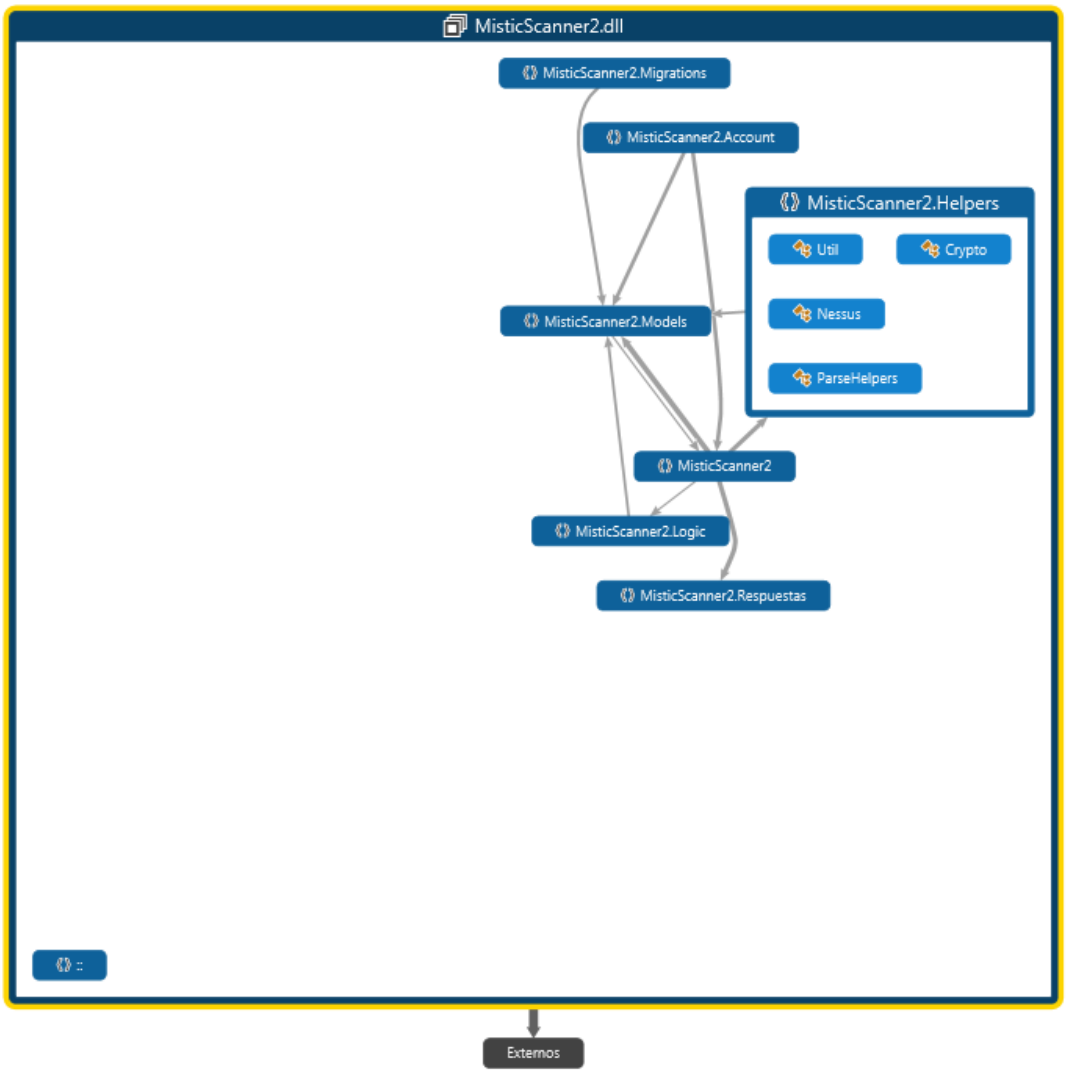

Figura 14

- Logic: Clases adicionales para la gestión de la lógica de negocio

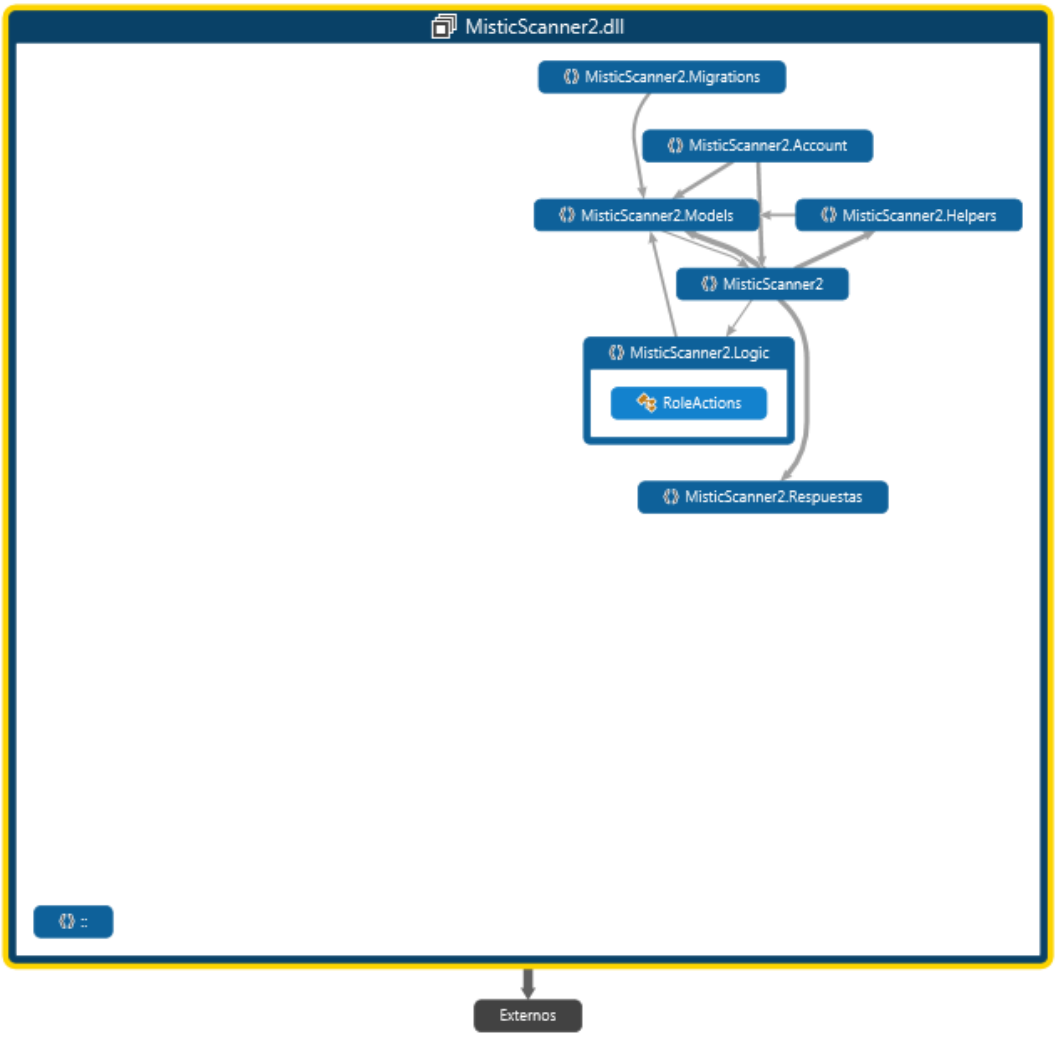

Figura 15

- Respuestas: Clases que representan el contenido de los mensajes de respuesta de Nessus, que son utilizadas para recabar la información que nos devuelve.

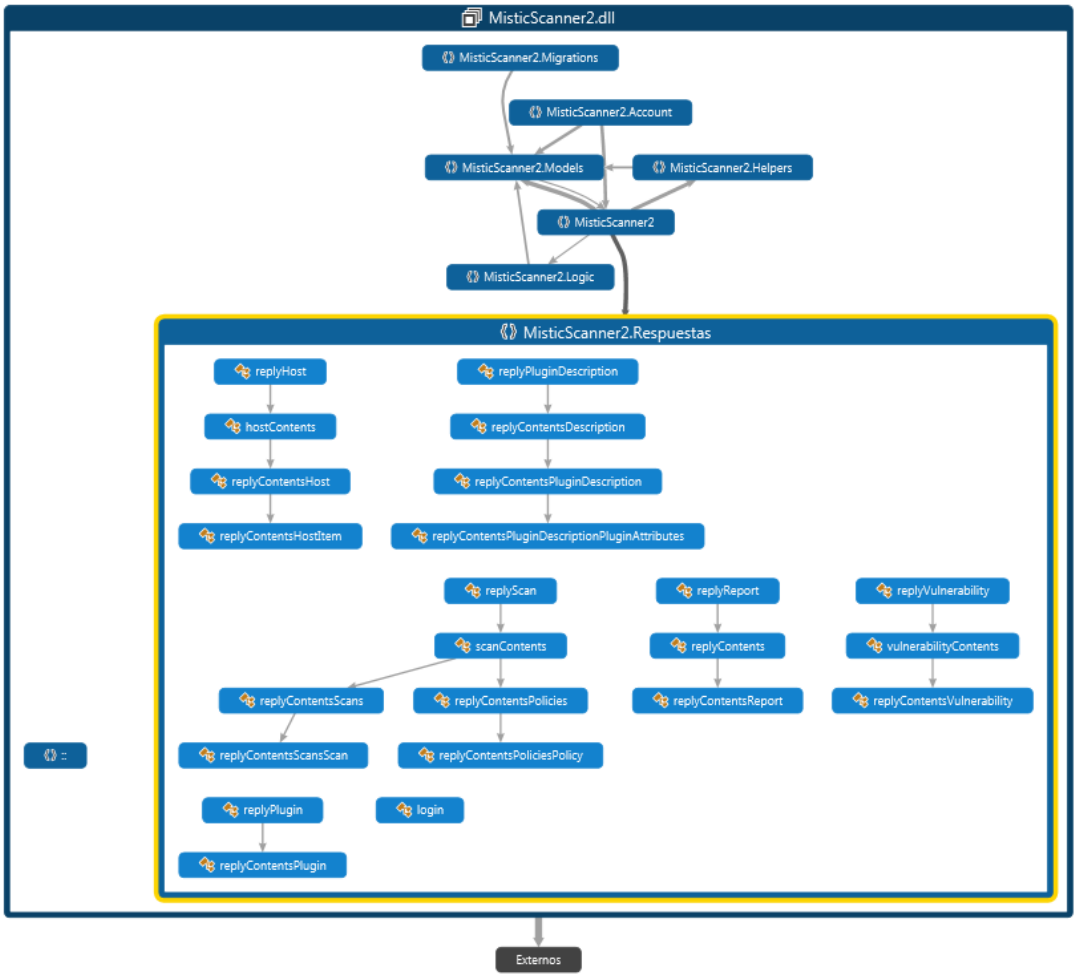

Figura 16

4. Comunicación con Nessus

La comunicación con Nessus requiere un apartado aparte ya que es una de las partes más importantes de la aplicación y la que tiene mayor novedad respecto a aplicaciones similares.
Para comunicarse con Nessus existen diversas opciones, línea de comandos, exportación de ficheros, etc. pero la que más nos interesó fue la interfaz de programación disponible mediante XML/RPC. A través de ella, mediante una simple petición web, obtenemos de respuesta un mensaje XML con la información de los escaneos de seguridad producidos en nuestros servidores, así como muchas más opciones que nos permiten casi controlar por completo nuestro servidor Nessus programáticamente.
Vamos a ver un ejemplo de cómo sería esta comunicación, utilizando fragmentos de código de la aplicación.
Si por ejemplo queremos obtener un listado de los informes presentes en nuestro sistema, bastaría con hace la petición:

respuesta = _nessus.Peticion(_nessus.Url + "/report/list", " ");

```
Respuestas.replyReport reply =
Helpers.ParseHelpers.ParseXML<Respuestas.replyReport>(respuesta);
```

Como podemos ver, hacemos una petición web a la url del servidor nessus mas “/report/list”, sin parámetros (hay peticiones que si los tienen) y se nos devuelve un cadena de texto en formato xml que nosotros deserializamos en una clase del tipo “replyReport”.

La petición web es una petición web normal salvo por el certificado y por una cookie de sesión que hemos obtenido previamente llamando a la dirección ” /login” con el usuario y contraseña del servidor Nessus como parámetros post

```
if (_tokenid == null)
        Login();

    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

    CookieContainer cookies = new CookieContainer();
    cookies.Add(_tokenid);
    request.CookieContainer = cookies;
    request.Method = "POST";

    string postData = param;
    byte[] byteArray = Encoding.UTF8.GetBytes(postData);
    request.ContentType = "application/x-www-form-urlencoded";
    request.ContentLength = byteArray.Length;
    Stream dataStream = request.GetRequestStream();
    dataStream.Write(byteArray, 0, byteArray.Length);
    dataStream.Close();

    HttpWebResponse response = (HttpWebResponse)request.GetResponse();

    dataStream = response.GetResponseStream();

    StreamReader reader = new StreamReader(dataStream);

    string responseFromServer = reader.ReadToEnd();
```

Como se puede ver, en la petición, lo primero que se hace es ver si estamos logados, y si no lo hacemos. A continuación se construye la petición POST con los parámetros si los hubiera y se recoge la respuesta en una cadena de texto para devolverla al cliente.

Mediante este interfaz XML/RPC podemos recoger información de Nessus, iniciar nuevos escaneos, consultar las políticas existentes, etc.

5. Conclusiones

Con este proyecto hemos aprendido la existencia de diversas formas de comunicación con diferentes escaneres de seguridad existentes en el mercado. Hemos elegido el que considerábamos más prometedor, la comunicación XML/RPC con Nessus, y hemos hecho un refresco a nuestros conocimientos de las tecnologías existentes para el desarrollo de aplicaciones web.

Hemos visto que no es fácil consolidar la información existente de vulnerabilidades en una empresa de tamaño medio, y que una aplicación que realice la tarea de almacenamiento y análisis de los mismos es altamente recomendable.

No hemos conseguido todos los logros marcados, ya que nos hubiera gustado aumentar el grado de interactividad con Nessus para recabar un mayor detalle de las vulnerabilidades presentes y un mayor trabajo de análisis y reporte sobre la información almacenada. El motivo principal ha sido la falta de tiempo, ya que no se estimó correctamente el trabajo a realizar en la etapa de desarrollo y la dificultad que entraña retomar el desarrollo de aplicaciones web con tecnologías novedosas que aunque son interesantes, implican un esfuerzo extra en el desarrollo

La planificación no se ha planteó correctamente, ya que habría que haber asignado más tiempo a la etapa de desarrollo y menos a la de análisis. Para conseguir acabar en el plazo dado ha habido que recortar en las funcionalidades previamente previstas.

Las líneas que han quedado pendientes serían dos:

- Ampliar la comunicación con Nessus a las funcionalidades que se consideraran interesantes y que todavía no estén implementadas
- Mejorar el sistema de análisis e informes sobre la información almacenada, ya que ahora mismo se tendría que hacer mediante consultas externas manuales de sql.

6. Glosario

- Nessus: Aplicación para la realización de escáneres de seguridad
- NMAP: Aplicación para la realización de escáneres de seguridad y más usos
- Metasploit: Aplicación para la realización de escáneres de seguridad, explotación de vulnerabilidades, etc.
- XML/RPC: Interfaz de programación remota mediante peticiones http y respuestas xml
- Web Forms: Tecnología de Microsoft para la realización de aplicaciones web
- MVC: Tecnología de Microsoft para la realización de aplicaciones web
- Model Binding: Tecnología de Microsoft para la sincronización del código y la base de datos
- ASP.NET: Tecnología de Microsoft para la realización de aplicaciones web
- SQL Server: Base de datos de Microsoft SQL Server
- Windows Azure: Tecnología en la nube de Microsoft

5. Bibliografía

- Nessus 5.0 REST Protocol Specification: Especificación del protocolo XML/RPC de comunicación con Nessus

  http://static.tenable.com/documentation/nessus_5.0_XMLRPC_protocol_guide.pdf

- OWASP Testing Guide V3.0: Guia de OWASP para el testeo de seguridad de aplicaciones web
  https://www.owasp.org/index.php/OWASP_Testing_Project

- Tutoriales de ASP.NET Web Forms: Para el desarrollo de aplicaciones web utilizando Web Forms
  http://www.asp.net/web-forms

6. Anexos