Universidad Tecnológica de El Salvador

FACULTAD DE INFORMÁTICA Y CIENCIAS APLICADAS

LICENCIATURA EN INFORMÁTICA

Tema:

“Seguridad de Aplicaciones Web en el Contexto Salvadoreño”

Presentado por:

Borja Moreno, Edgar Ernesto

Guevara Mejía, Cesar David

Para optar al grado de:

Licenciatura en Informática

Septiembre 2014

San Salvador, El Salvador

AUTORIDADES

ING. NELSON ZÁRATE SÁNCHEZ

RECTOR

LIC. JOSE MODESTO VENTURA

VICERRECTOR ACADÉMICO

ING. FRANCISCO ARMANDO ZEPEDA

DECANO

JURADO EXAMINADOR:

ING. VICENTE ANTONIO ZARCEÑO JACO

PRESIDENTE

ING. OSCAR ERNESTO RODRÍGUEZ ALFARO

PRIMER VOCAL

ING. ÁLVARO ERNESTO ANGULO VIOLANTES

SEGUNDO VOCAL

SEPTIEMBRE, 2014

SAN SALVADOR, EL SALVADOR, CENTROAMÉRICA

Universidad Tecnológica de El Salvador

ACTA DE EXAMEN PROFESIONAL

HABIÉNDOSE REUNIDO EL JURADO CALIFICADOR INTEGRADO POR:

Ing. Oscar Ernesto Rodriguez Alfaro, Ing. Álvaro Ernesto Angulo Violantes, Ing. Vicente Antonio Zarceño Jaco, a las 12:00p.m. del día Sábado, 31 de Mayo de dos mil catorce.

Y LUEGO DE HABER DELIBERADO SOBRE EL EXAMEN PROFESIONAL DE LOS ALUMNOS:

1- César David Guevara Mejía CARNET 17-3227-2004

2- Edgar Ernesto Borja Moreno CARNET 17-2897-2002

QUIENES PRESENTARON DEFENSA DE SU TRABAJO DE GRADUACION TITULADO:

"Seguridad de Aplicaciones Web en el Contexto Salvadoreño"

PARA OPTAR AL GRADO DE: **LICENCIATURA EN INFORMATICA**

Y DEL CUAL TAMBIEN EVALUARON LOS CONOCIMIENTOS RELACIONADOS CON EL TEMA DEL MISMO

POR LO QUE ESTE JURADO RESUELVE DECLARAR EL EXAMEN COMO:

APROBADO.

YA QUE CUMPLE CON LOS REQUISITOS ESTABLECIDOS EN EL REGLAMENTO DE GRADUACION DE LA UNIVERSIDAD.

San Salvador, 31 de mayo de dos mil catorce.

F. _____
PRIMER VOCAL
Ing. Oscar Ernesto Rodriguez Alfaro

F. _____
SEGUNDO VOCAL
Ing. Álvaro Ernesto Angulo Violantes

F. _____
PRESIDENTE DEL JURADO
Ing. Vicente Antonio Zarceño Jaco

AGRADECIMIENTOS

Agradezco a Dios, quien me dio la sabiduría y fuerza de llegar a este punto de mi carrera, a mi esposa Inga y a mis padres Edgar y Yanira, quienes me apoyaron en todo momento.

También agradezco a los catedráticos y compañeros de la Universidad de quienes pude aprender en estos años.

Edgar Borja.

AGRADECIMIENTOS

Al creador de todo lo que existe, por haberme bendecido con la oportunidad de emprender este camino, el cual no ha sido fácil ni mucho menos corto, pero si ha estado lleno de muchas bendiciones.

A mis padres Gonzalo y Blanca, por enseñarme a seguir adelante y alentarme a siempre luchar por mis sueños, por su apoyo y cariño incondicional.

A mi esposa Alejandra, por su inmensa comprensión, porque siempre está en el momento necesario para levantar mis ánimos.

César Guevara

Contenido

Introducción

Seguridad de aplicaciones web y sitios web, sin duda es un tema que encierra muchas preguntas, en los últimos años ha llamado la atención de muchas empresas e incluso gobiernos, ya que conforme pasa el tiempo y el desarrollo informático es más avanzado, hay más información expuesta y es de suma importancia el garantizar que esa información sea gestionada por las personas a las que está destinada. El Salvador no es la excepción, junto con los avances tecnológicos cada día hay más empresas que utilizan internet para facilitarle acceso a sus clientes, como bancos, universidades, empresas distribuidoras, el gobierno, etc., esto representa grandes ventajas para las empresas al ahorrar en recurso humano o en gastos de atención al cliente, y para los clientes es una gran ventaja el hacer las gestiones desde la comodidad de su casa o trabajo.

Es a raíz de esto que nos preguntamos si ¿En El Salvador las instituciones que prestan sus servicios a través de la web han tomado en cuenta la seguridad?, si los bancos han tomado en cuenta las posibles vulnerabilidades para garantizarle al cliente que sus transacciones electrónicas que impliquen o no dinero estén seguras.

La investigación se centra en identificar todos los sitios que estén registrados como Salvadoreños (terminación .sv), identificar qué tipo de tecnología utilizan en su programación y qué servidor web es utilizado para prestar el servicio,

identificando al mismo tiempo si las preferencias de las instituciones Salvadoreñas al elegir los servidores para sus servicios están en El Salvador o en el extranjero.

Logrando con esto hacer un diagnóstico aproximado a la realidad de nuestro país en relación a las buenas prácticas en el desarrollo de aplicaciones web, identificando la preferencia hacia las diferentes plataformas tecnológicas ya sean de pago o de no pago.

Capítulo I: Planteamiento del problema

Las aplicaciones y sitios web son una herramienta necesaria para cualquier empresa u organización, estas herramientas contienen información clave y son la carta de presentación de la organización.

Si la información o las transacciones que se realizan a través de estas aplicaciones fueran comprometidas por un atacante las consecuencias pueden ir desde afectar la imagen de la empresa, hasta un colapso financiero o la pérdida de capital intelectual; pero también los clientes que confiaron en las medidas de seguridad implementadas por la empresa pueden verse críticamente afectados, pues pueden sufrir un fraude de crédito o hasta un robo de identidad.

Existen múltiples vectores de ataque a los que cualquier organización con presencia en Internet está expuesta; en el presente documento se exponen los principales riesgos, acciones de mitigación aplicables, y en base a la información disponible para El Salvador, se presenta un análisis del estado de la seguridad de las aplicaciones y sitios web enfocándose en la vulnerabilidad por el uso de componentes desactualizados y comparando la seguridad preventiva en base al país donde está basado cada servidor.

1.1 Objetivos

1.1.1 Objetivo general

Describir los principales riesgos de seguridad a los que están expuestos los sitios y aplicaciones web salvadoreños publicados en Internet, determinando el uso de prácticas de seguridad preventiva utilizados.

1.1.2 Objetivos específicos

- Clasificar los principales riesgos y amenazas que enfrentan los sitios y aplicaciones web.
- Analizar el estado de seguridad preventiva con el uso de componentes actualizados para todos los sitios de dominio .sv en base al país donde se encuentran los servidores.
- Describir a través de un caso de estudio el proceso de resolución que debe seguir una empresa en el momento que un problema de seguridad en un sitio o aplicación web es detectado.

1.2 Alcance

Para los propósitos de esta investigación, se han incluído los principales riesgos de seguridad informática presentes en la web salvadoreña como parte del desarrollo y arquitectura en la capa de la aplicación, considerando los 10 riesgos más comunes en la actualidad de acuerdo a estudios internacionales (OWASP Foundation, 2013), y se excluyen los riesgos y amenazas que pueden generarse por los medios de comunicación como redes, protocolos, DNS, o por los métodos de almacenamiento de datos como técnicas de encriptación o unidades físicas de almacenamiento, también se excluyen las amenazas que requieren el uso de ingeniería social o la interacción personal y física de un atacante con las potenciales víctimas.

1.3 Beneficiarios

Con esta investigación se pretende beneficiar a diferentes actores dentro de nuestra sociedad, entre de los cuales podemos mencionar

- Estudiantes: que al tener conocimiento de estos factores de riesgo, les proporcionara ventaja competitiva en el desarrollo profesional.
- Universidad: que al estar consciente de los muchos riesgos, podrá tener un parámetro real para incluir esta temática en el pensum.
- Usuarios: que al estar al tanto de las amenazas, serán más cuidadosos en relación a los sitios en los que ingresan información confidencial.

Capítulo II: Marco conceptual

Los atacantes pueden usar muchas rutas diferentes a través de una aplicación o sitio web para hacer daño a una empresa u organización. Cada una de estas rutas representa un riesgo que puede, o no, ser lo suficientemente grave como para merecer atención.

A veces, estos caminos son fáciles de encontrar y explotar y a veces son muy difíciles. Del mismo modo, el daño que se causa puede ser de mínimas consecuencias, o puede hacer colapsar el negocio o empresa. Para determinar el riesgo al que está expuesta la organización es posible evaluar la probabilidad asociada a cada agente de amenaza, de ataque, y debilidad en la seguridad y combinarlo con una estimación del impacto técnico y empresarial a una organización. En conjunto, estos factores determinan el riesgo general.

A continuación se describen los riesgos más comunes existentes en aplicaciones y sitios web, y los pasos recomendados para su mitigación.

2.1 Vulnerabilidades de inyección

Las vulnerabilidades de inyección, tales como de SQL, del sistema operativo, y la inyección LDAP se producen cuando datos que no son de confianza se envían a un intérprete como parte de un comando o una consulta. Los datos hostiles del atacante puede engañar al intérprete para que ejecute comandos no deseados u otorgue acceso a los datos sin la debida autorización .

La mejor manera de averiguar si una aplicación es vulnerable a la inyección es verificar que toda aplicación que use intérpretes separe claramente los datos no confiables del comando o consulta. Para las llamadas SQL, esto significa utilizar variables cuyo contenido sea estrrictamente analizado para todas las sentencias preparadas y procedimientos almacenados, y evitar las consultas dinámicas, donde las llamadas SQL son construídas utilizando entradas de los usuarios.

Es necesario comprobar el código, esta es una forma rápida y precisa para ver si la aplicación utiliza intérpretes de forma segura. Existen herramientas de análisis de código que pueden ayudar a un analista de seguridad de encontrar las instancias donde se han usado intérpretes y rastrear el flujo de datos a través de la aplicación. Las pruebas de penetración pueden validar estos temas a través de la elaboración de análisis que confirmen la existencia de vulnerabilidades.

El escaneo dinámico automatizado que se ejecuta a través de una aplicación especializada llamada analizador puede ayudar a determinar si existen vulnerabilidades de inyección explotables. Estos analizadores no siempre pueden llegar a los intérpretes existentes en la aplicación y pueden tener dificultades para detectar si un ataque fue exitoso. El mal manejo de errores hace que los errores de inyección sean más fáciles de descubrir, ya que la existencia de una excepción o error expuesto al usuario que se genera a través de un intérprete es un indicador claro de que ese segmento de la aplicación no fue desarrollado con las consideraciones correctas y puede ser vulnerable a inyección.

Para prevenir la inyección se requiere mantener los datos no confiables separados de comandos y consultas.

La opción preferida es usar una API (del inglés Application Programming Interface) de seguridad que evite el uso del intérprete completamente o proporcione una interfaz con parámetros. También es necesario considerar que las API, o los procedimientos almacenados, que son parametrizados, aún pueden permitir la inyección si no se implementan apropiadamente, ya que parte de los comandos y consultas generados pueden provenir de una entrada generada por el usuario.

Si una API que funcione a través de parámetros no está disponible, debemos escapar cuidadosamente caracteres especiales utilizando la sintaxis de escape específica para ese intérprete.

La validación positiva de entrada, también conocida como “lista blanca” o “white list” también se recomienda, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en su ejecución. Si una aplicación requiere el uso de caracteres especiales en los intérpretes, será necesario recurrir a una API como se describe en los párrafos anteriores. (OWASP Foundation, 2013)

2.2 Fallas de autenticación y gestión de sesiones

Las funciones de una aplicación relacionadas con la autenticación y gestión de sesiones con frecuencia pueden contener errores de implementación, tal y como se presenta en el caso de estudio examinado en el presente documento, esto permite a los atacantes comprometer contraseñas, o tokens de sesión, o explotar otras fallas de implementación que permiten asumir la identidad de otros usuarios.

Los activos de administración de sesiones, como las credenciales de usuario y los identificadores de sesión pueden ser protegidos adecuadamente, sin embargo existen múltiples variables que pueden incrementar el riesgo al que está expuesta una aplicación. A continuación se listan algunos de estas variables:

- Las credenciales de autenticación de usuario que no se protegen utilizando hash o cifrado a la hora de almacenarse en una base de datos.

- Las credenciales que pueden ser descubiertas o sobreescritas mediante funciones vulnerables en la administración de cuentas (por ejemplo, la creación de cuentas, cambio de contraseña, recuperación de contraseñas, la debilidad de identificadores de sesión).

- Los identificadores de sesión que se exponen en la dirección URL son vulnerables a ataques especializados de reescritura de URL.

- Los identificadores de sesión vulnerables a ataques de fijación de sesión, donde las variables que identifican sesiones existentes de los usuarios pueden ser manipuladas por los usuarios, ya sea en cookies residentes en el navegador, o en variables del servidor que se establecen a través de datos ocultos enviados desde formularios.

- Los identificadores de sesión no caducan, o sesiones de usuario o tokens de autenticación, (SSO) fichas de inscripción en particular individuales, no se invalidan correctamente durante la desconexión.

- Identificadores de sesión que no expiran después de inicio de sesión correcto, lo que permite que sesiones establecidas anteriormente queden abiertas por un tiempo indefinido, desde computadoras a las que el usuario podría ya no tener acceso.

- Contraseñas, identificadores de sesión y otras credenciales que se envían a través de conexiones no encriptadas, tanto al momento que el usuario inicia sesión con el servidor, como durante procesos de gestión de cuentas como reestablecimiento de contraseñas o envío de correos de confirmación para la creación de cuentas.

La recomendación principal para una organización es poner a disposición de los desarrolladores un conjunto de herramientas y estándares que les permita trabajar con eficiencia y de manera segura, por ejemplo:

- Un único conjunto de controles de autenticación y gestión de sesiones que se utilice a través de toda la organización; estos controles deberán cumplir con los requisitos de seguridad de la industria, y deberán ser el estándar para todas las gestiones de usuarios.

- Tener una interfaz sencilla para los desarrolladores, a través de un API de autenticación y gestión de usuarios, complementado por un manual para el desarrollador y una biblioteca con buenos ejemplos de su utilización sobre los cuales ellos puedan trabajar.

- Deben hacerse grandes esfuerzos para evitar fallas de XSS que pueden ser utilizados para robar los identificadores de sesión.

2.3 Fallas de XSS (secuencias de comandos en sitios cruzados)

Las fallas de XSS (Cross-site scripting o secuencias de comandos en sitios cruzados) ocurren cuando una aplicación toma datos no confiables y los envía a un cliente a través de un navegador web sin necesidad de una validación adecuada o sin escapar los caracteres especiales. El XSS permite a los atacantes ejecutar scripts en el navegador de la víctima, estos scripts se pueden usar para secuestrar sesiones de usuario, desfigurar sitios web, o redirigir al usuario a sitios maliciosos .

Una aplicación es vulnerable si no se asegura para que todas las entradas proporcionadas por el usuario sean escapadas correctamente, o si no se verifica

que la entrada del usuario es segura a través de la validación de su contenido, antes de incluir esa entrada en la página de salida. Sin el mecanismo de escape adecuado o validación, estas aportaciones serán tratadas como el contenido activo en el navegador. Si se utiliza Ajax para actualizar dinámicamente la página, la aplicación deberá de hacerlo a través de un API de JavaScript seguro. Para los API de JavaScript no seguros, también se debe utilizar codificación o validación.

Con herramientas automatizadas pueden detectarse algunos problemas de XSS de forma automática. Sin embargo, una aplicación que genera páginas de salida y utiliza diferentes intérpretes secundarios en el navegador como JavaScript, ActiveX, Flash y Silverlight hace difícil la detección automática. Por lo tanto, para estar completamente seguros, se requiere una combinación de revisión de código manual y pruebas de penetración, además de los enfoques automatizados.

Las tecnologías Web 2.0, como Ajax, hacen el XSS mucho más difícil de detectar a través de herramientas automatizadas, ya que las páginas web cambian su estado y estructura dependiendo de las acciones de los usuarios, lo que crea una serie de combinaciones y secuencias posibles que no pueden estar completamente al alcance de una herramienta automatizada.

Para prevenir el XSS se requiere la separación de datos no confiables del contenido activo del navegador.

La opción preferida es la de escapar correctamente todos los datos que no son de confianza y mantenerlos separados del contexto HTML (cuerpo, atributos, JavaScript, CSS, o URL). Existen diversas técnicas para escapar datos antes de ser presentados en una página web, las más comunes son la eliminación de los caracteres especiales, etiquetas y referencias a recursos externos que pueden ser parte de la entrada de un usuario en un campo de texto, área de texto o en la URL, o el reemplazo de las entradas mencionadas anteriormente por secuencias de escape propias de HTML, JavaScript o del lenguaje de programación utilizado en el servidor.

También se recomienda la validación de entrada por “lista blanca” o “white list”, ya que ayuda a proteger contra el XSS, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en las entradas de los usuarios. Esta validación debe, en lo posible, validar la longitud, caracteres, el formato y las reglas de negocio para los datos antes de aceptar la entrada.

Para el contenido rico, que incluye archivos multimedia y otros recursos como XML, o SVG, es necesario evaluar la utilización de bibliotecas de auto-desinfección como AntiSamy o Project Desinfectant en Java para HTML. (OWASP Foundation, 2013)

2.4 Referencias directas no seguras a objetos

Una referencia directa ocurre cuando un desarrollador expone una referencia a un objeto interno de la aplicación, como un archivo, un directorio o base de datos. Sin un control de accesos o de otro tipo de protección, los atacantes pueden manipular esas referencias para acceder a datos no autorizados.

La mejor manera de averiguar si una aplicación es vulnerable a las referencias directas no seguras a objetos es verificar que todas las referencias a objetos tengan las defensas adecuadas. Para ello, es necesario tomar en cuenta las siguientes consideraciones:

- Para las referencias directas a los recursos restringidos, la aplicación debe de verificar que el usuario está autorizado a acceder al recurso exacto en el que ha solicitado. Estas referencias pueden existir en la aplicación como enlaces de descarga de archivos, hipervínculos, archivos adjuntos que pueden obtenerse de la aplicación por correo electrónico, etc.

- Si la referencia es una referencia indirecta, la asignación a la referencia directa debe limitar los valores a los autorizados para el usuario actual; esta situación se presenta cuando las descargas de la aplicación se administran a través de un script que funciona como gateway o puerta de

enlace hacia los objetos, el gateway recibe parámetros de la aplicación o del usuario sobre cuál objeto debe ser descargado. El gateway siempre debe verificar si el usuario activo tiene permisos para realizar la acción solicitada sobre el objeto relevante.

La revisión del código de la aplicación se puede realizar rápidamente con cualquiera de los dos enfoques, referencias directas o indirectas, siempre que se implementen de manera segura. Las herramientas automatizadas típicamente no buscan ese tipo de errores porque no pueden reconocer cuando se requiere la protección de ciertos recuros o si las vías de acceso a los recursos son seguras o inseguras.

La prevención de referencias directas no seguras a objetos requiere la selección de un enfoque para la protección de cada objeto accesible para el usuario (por ejemplo, número de objeto, identificador único, nombre de archivo).

Una forma de prevenir estas vulnerabilidades es usar referencias indirectas a objetos limitadas por usuario o sesión. Esto evita que los atacantes se orientan directamente a recursos no autorizados. Por ejemplo, en lugar de utilizar los identificadores de los recursos tal y como se usan en la base de datos, los identificadores se limitan a una lista desplegable que sólo muestra aquellos autorizados para el usuario, y se identifican con los números del 1 al 6, de esa manera el usuario no tiene conocimiento de o forma de referenciar los objetos a

los que no debería tener acceso. La aplicación debe asignar la referencia indirecta limitada del ámbito del usuario de nuevo a la llave de la base de datos real en el servidor.

Existen APIs que incluyen mapas de referencias de acceso tanto secuenciales como aleatorias que los desarrolladores pueden utilizar para eliminar las referencias directas a objetos.

Otro requisito para eliminar las vulnerabilidades de referencias directas a objetos es que cada uso de una referencia desde una fuente no confiable debe incluir un control de accesos para garantizar que el usuario está autorizado para acceder al objeto solicitado.

2.5 Configuración de seguridad incorrecta

Una buena seguridad requiere tener una configuración segura definida y en uso para aplicaciones, frameworks, servidores de aplicaciones, servidores web, servidores de base de datos, y la plataforma del sistema operativo. Los ajustes seguros deben definirse, implementarse y mantenerse, ya que los valores por defecto pueden ser inseguros. Además, el software debe mantenerse actualizado.

La configuración de un sistema para máxima seguridad se conoce como “Hardening” o “Endurecimiento”, el proceso de endurecimiento incluye:

- Actualización del software. Esto incluye el sistema operativo, servidor web, servidor de aplicaciones, servidor de bases de datos, las aplicaciones y todas las bibliotecas de código.

- La deshabilitación de las características u opciones innecesarias que puedan estar instaladas, por ejemplo, puertos, servicios, páginas, cuentas o privilegios.

- La modificación de cuentas predeterminadas y sus contraseñas.

- Prevenir que los mensajes de error que se muestren al usuario puedan revelar trazas de pila (stack traces) u otro tipo de información interna del sistema que permitan una ventana al funcionamiento interno de éste.

- Las configuraciones de seguridad en los frameworks (por ejemplo, Struts, Spring, ASP.NET) y las bibliotecas deben establecerse en un nivel de seguridad alto.

Si no se aplica un proceso de configuración de seguridad correcto a la aplicación, los sistemas corren un riesgo más alto.

A continuación se presentan una serie de recomendaciones que pueden establecerse para mejorar la configuración de seguridad:

- Establecer un proceso de endurecimiento repetible que haga que sea rápido y fácil instalar otro entorno que esté protegido correctamente. Los entornos de desarrollo, control de calidad, producción y otros deben ser configurados de manera idéntica (con diferentes contraseñas utilizadas en cada ambiente). Este proceso debe ser automatizado para reducir al mínimo el esfuerzo necesario para configurar un nuevo entorno de seguridad.

- Crear un proceso para estar actualizados con todos los nuevos parches y actualizaciones de software de forma oportuna para cada entorno implementado. Esto debe incluir también a todas las bibliotecas de código, tanto propias como de terceros.

- Diseñar una arquitectura de aplicación sólida que proporcione la separación segura y eficaz entre los componentes, ya que de esa forma, en caso de detectarse un problema de seguridad, es posible aislarlo rápidamente y aplicar los cambios necesarios sin afectar a los otros componentes.

- La ejecución de análisis y auditorías periódicas con el propósito de detectar configuraciones incorrectas o parches faltantes.

2.6 Exposición de datos sensibles

Muchas aplicaciones web no protegen adecuadamente los datos sensibles, tales como tarjetas de crédito, nombres completos, información personal, identificaciones fiscales, así como las credenciales de autenticación . Los atacantes pueden robar o modificar dichos datos débilmente protegidos para llevar a cabo el fraude de tarjetas de crédito, extorsiones, robo de identidad u otros delitos. Los datos sensibles merecen una protección adicional, como la encriptación en reposo o durante el transporte, así como precauciones especiales cuando se intercambian con el navegador .

Lo primero que hay que determinar es qué datos son lo suficientemente sensibles como para requerir protección adicional. Por ejemplo, contraseñas, números de tarjetas de crédito, registros de salud, e información personal deben ser protegidos. Para todos esos datos es necesario evaluar los siguientes vectores de riesgo:

- Datos almacenados en texto plano y en dispositivos de largo plazo, incluyendo las copias de seguridad.

- Datos que se transmiten en texto plano, ya sea interna o externamente. Hay que ser especialmente cuidadosos con el tráfico que se envía a través de Internet.

- La utilización de algoritmos criptográficos antiguos o en los que se hayan encontrado vulnerabilidades.

- La existencia de claves criptográficas débiles, la falta de una gestión apropiada para las claves, incluyendo su rotación periódica.

- La ausencia de directivas de seguridad del navegador o cabeceras (headers) recomendadas cuando los datos sensibles son proporcionados por o enviados al navegador. Esto incluye los encabezados que indiquen al proxy y al navegador que no se deben almacenar en el cache local copias de las páginas que contienen información sensible.

Los riesgos generados por el mal uso o configuración de herramientas criptográficas está fuera del alcance de este documento, pero es importante mencionar que estas vulnerabilidades de las capas inferiores del modelo OSI compromenten la seguridad de los datos y de las aplicaciones de forma grave, como se experimentó en fechas recientes cuando se reportó que más de dos terceras partes de los sitios web más grandes del mundo fueron afectados por la vulnerabilidad de OpenSSL bautizada como Heartbleed.

A continuación se presentan las recomendaciones de seguridad para prevenir la exposición de datos sensibles:

- Teniendo en cuenta las amenazas a las que pueden estar expuestas estos datos, ya sea por potenciales ataques internos, o por usuarios externos, es necesario asegurarse de encriptar todos los datos confidenciales en reposo y durante el transporte de una manera que estén protegidos.

- No almacenar los datos confidenciales de forma innecesaria. Éstos deben de ser desechados tan pronto como sea posible. Si los datos no están en la base de datos no pueden ser robados.

- Usar consistentemente algoritmos estándares fuertes y utilizar claves de alta seguridad, con un proceso de gestión adecuado para las mismas. Existen estándares internacionales como el FIPS 140 para la validación de módulos criptográficos.

- Asegurar las contraseñas almacenanadas con un algoritmo diseñado específicamente para la protección de contraseñas, como bcrypt PBKDF2 o scrypt.

- Desactivar la función de autocompletar en los formularios de recolección de datos sensibles y prevenir el almacenamiento en caché para las páginas que contienen datos confidenciales.

2.7 Falta de control de accesos a nivel de función

La mayoría de las aplicaciones web verifican los derechos de acceso a nivel de función antes de que cada una de las funciones sea visible en la interfaz del usuario, sin embargo, las aplicaciones necesitan realizar las mismas comprobaciones de control de acceso en el servidor cuando se accede a cada función, ya que un atacante puede alterar las peticiones o los envíos de datos al servidor para ejecutar funciones o realizar llamadas que no están disponibles en la interfase de usuario. Si no se verifican las solicitudes entrantes, los atacantes pueden ser capaces de forjar las solicitudes con el fin de acceder a las funciones sin la debida autorización .

La mejor manera de averiguar si una aplicación puede restringir el acceso de manera adecuada a nivel de función es verificar todas las funciones de aplicación. La siguiente lista muestra una serie de indicadores que ayudan en la identificación de riesgos por falta de control de accesos a nivel de función:

- Si la interfase de usuario de la aplicación muestra funciones a las que el usuario no debería de tener acceso.

- La falta de autenticación o autorización de acciones del lado del servidor.

- Si los controles del lado del servidor ejecutan procedimientos basados sólamente en la información proporcionada por el atacante.

Una técnica que permite detectar estas vulnerabilidades es el uso de un proxy, de esta forma, es posible navegar por la aplicación utilizando una cuenta menos privilegiada, luego se debe intentar acceder a páginas restringidas para el rol. Si las respuestas del servidor son iguales a las obtenida en la navegación normal, la aplicación es probablemente vulnerable.

También es posible comprobar la aplicación de controles de acceso en el código. Para esto es necesario seguir una sola solicitud de privilegio a través del código y verificar el patrón de autorización, a continuación, nos basamos en el código fuente para encontrar los procedimientos que no se está siguiendo ese patrón de autorización para cada solicitud.

Las herramientas automatizadas no suelen encontrar estos problemas.

La aplicación debe tener un módulo de autorización consistente y fácil de analizar que se invoque desde todos los procedimientos de negocio. Con frecuencia, esta protección es proporcionada por uno o más componentes externos al código de

la aplicación, ya que muchos ambientes corporativos cuentan con controladores de dominio de LDAP o Active Directory.

Las siguientes prácticas reducen la posibilidad de que la falta de control de acceso a nivel de función se vuelva una amenaza para las aplicaciones:

- Hacer un análisis del proceso de gestión de permisos y asegurarse de que puede actualizar y auditar fácilmente. El código debe ser fácil de leer.

- Las aplicaciones deben negar todo acceso de forma predeterminada, los permisos adicionales que sean necesarios deben ser requeridos explicitamente por la funciones.

- Si la función está involucrada en un flujo de trabajo, debe verificarse que que el estado actual del flujo es el adecuado para permitir el acceso solicitado.

La mayoría de las aplicaciones web no muestran enlaces o botones para funciones no autorizadas, pero este “control de acceso a nivel de presentación” en realidad no ofrece protección. También deben implementarse controles en el controlador o la lógica de negocio. (Riden & McGeehan)

2.8 Falsificación de petición en sitios cruzados

Un ataque CSRF (cross-site request forgery o falsificación de petición en sitios cruzados) fuerza al navegador de la víctima que ha iniciado sesión en la aplicación a enviar una petición HTTP forzada a un sitio externo, esta petición incluye automáticamente la cookie de sesión de la víctima o cualquier otra información de autenticación de una aplicación web vulnerable. El CSRF también habilita al atacante a obligar al navegador de la víctima para generar peticiones que la aplicación vulnerable interpreta como peticiones legítimas de la víctima.

Comprobar si una aplicación es vulnerable requiere identificar si alguno de los vínculos o formularios carecen de un token CSRF impredecible. Sin un token así, los atacantes pueden forjar peticiones maliciosas. Una defensa alternativa es exigir al usuario que confirme su intención de presentar la solicitud, ya sea a través de reautenticación, o alguna otra prueba de que es un usuario real (por ejemplo, un CAPTCHA).

Es importante concentrarse en los vínculos y los formularios que invocan funciones de cambio de estado, como actualizaciones a la cuenta del usuario o a los elementos de datos de la aplicación, ya que esos son los objetivos más importantes del CSRF.

Se deben verificar también las transacciones de varios pasos, ya que no están necesariamente excentas de esta vulnerabilidad. Los atacantes pueden falsificar fácilmente una serie de peticiones mediante el uso de varias etiquetas o posiblemente de JavaScript.

Necesitamos tener en cuenta que las cookies de sesión, direcciones IP de origen, y otra información que se envía de forma automática por el navegador no podrá servir de defensa contra CSRF ya que esta información también se incluye en las solicitudes falsas.

La prevención de CSRF por lo general requiere la inclusión de un token impredecible en cada solicitud HTTP. Estas fichas deben, como mínimo, ser únicas para cada sesión de usuario.

La opción preferida es incluir el token único en un campo oculto. Esto hace que el valor que se envíe en el cuerpo de la solicitud HTTP, y se previene su inclusión en la URL, que es más propensa a estar expuesta.

El token único también se puede incluir en la dirección URL, o en un parámetro de la URL. Sin embargo, aplicarlo de esta manera conlleva un mayor riesgo de que el URL sea expuesto a un atacante, comprometiendo así el token secreto.

Requerir al usuario que vuelva a autenticarse, o demostrar que es un usuario y no un script automatizado (por ejemplo, a través de un CAPTCHA de) también puede proteger contra el CSRF.

2.9 Re-direccionamientos no validados

Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios a otras páginas y sitios web, y pueden usar datos no confiables para determinar las páginas de destino. Sin una correcta validación, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o de malware, o utilizar las redirecciones para acceder a páginas no autorizadas.

Las mejores maneras de averiguar si una aplicación tiene cualquier redirección o reenvíos no validados son:

- Revisar el código para todos los usos de redireccionamiento o reenvíos llamados transferencia en .NET, a través de los métodos Request.Redirect y Server.Transfer respectivamente. Para cada uso, se debe comprobar si la dirección URL de destino está incluida en los valores de los parámetros. Si es así, ese procedimiento es vulnerable a este tipo de ataque; también si la URL no se valida contra una lista blanca, es vulnerable.

- Otro procedimiento útil es usar una araña HTTP (spider) para recorrer todos los enlaces y páginas del sitio y ver si genera cualquier redirección (códigos de respuesta HTTP 300 a 307, normalmente 302). Al revisar los parámetros proporcionados antes de la redirección se debe examinar si parecen ser parte de la dirección URL de destino o una porción del enlace. Si es así, se debe hacer una prueba alterando el parámetro con una URL diferente si se debe observar si el sitio redirige a la nueva dirección.

- Si el código no está disponible, se deben de comprobar los parámetros enviados para ver si parecen ser parte de una redirección o URL de destino y probar los que sí lo parecen.

Las vulnerabilidades en el uso de las redirecciones y reenvíos se pueden limitar de varias maneras:

- Simplemente evitar el uso de redirecciones y reenvíos.

- Si se utilizan, no se deben aplicar parámetros proporcionados por el usuario en el cálculo del destino.

- Si no es posible evitar el uso de parámetros de destino, se debe confirmar que el valor proporcionado es válido y autorizado para el usuario.

Se recomienda que cualquiera de los parámetros de destino debe ser un valor interno de la aplicación, en lugar de la URL real o parte de la URL, y que el

código del lado del servidor traduzca este valor a la dirección URL de destino real.

Evitar estos defectos es muy importante ya que son un blanco favorito de los phishers que buscan ganarse la confianza del usuario.

2.10 Uso de componentes con vulnerabilidades conocidas

Los componentes, como las bibliotecas, los frameworks (marcos de trabajo), y otros módulos de software, casi siempre se ejecutan con privilegios completos. Si se explota un componente vulnerable, un ataque de este tipo puede facilitar la pérdida de datos importantes o la toma de control del servidor. Las aplicaciones que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de la aplicación y permitir un amplio rango de posibles ataques e impactos.

En teoría, es fácil de averiguar si la aplicación o ambiente están usando cualquier componente o biblioteca vulnerables. Desafortunadamente, los reportes de vulnerabilidades de software comercial o de código abierto no siempre especifican exactamente qué versiones de un componente son vulnerables en

una forma estándar y fácil de buscar. Además, no todas las bibliotecas utilizan un sistema de numeración de versiones comprensible.

Lo peor de todo no todas las vulnerabilidades que se reportan a una entidad central donde es fácil realizar búsquedas, sino aquellas que se mantienen ocultas por parte de quien las haya descubierto o por parte del fabricante del componente que contiene la falla.

Determinar si una aplicación o un ambiente es vulnerable requiere estar al día con las listas y anuncios publicadas por los fabricantes de software, y de esa forma corregir cualquier falla que pueda ser un vector de ataque. Si uno de los componentes utilizados tiene una vulnerabilidad, se debe evaluar cuidadosamente si la aplicación es realmente vulnerable mediante una verificación para ver si se utiliza la parte del componente de que incluye la vulnerabilidad y si la falla genera un impacto que sea de interes para potenciales atacantes.

Una opción es no utilizar componentes que no fueron programados por la organización. Pero esto no es muy realista.

La mayoría de productos de software utilizan componentes que no tienen parches para vulnerabilidades en las versiones antiguas. En lugar de esto, la mayoría de fabricantes simplemente arreglan el problema en la próxima versión,

así que la actualización hacia las nuevas versiones es crítica. Los proyectos de software deben tener un proceso establecido para:

- Identificar todos los componentes y las versiones que se estén usando, incluyendo todas las dependencias. (por ejemplo, las versiones de plugins requeridos del navegador, bibliotecas de controles web, etc).

- Velar por la seguridad de estos componentes en las bases de datos públicas donde se reportan vulnerabilidades descubiertas, listas de correo del fabricante, y las listas de correo de seguridad informática, todo con el propósito de mantener los componentes actualizados.

- Establecer políticas de seguridad que rigen el uso de componentes externos, como la exigencia de ciertas prácticas de desarrollo de software por parte de los fabricantes, la existencia de pruebas de seguridad, y el uso de estándares internacionales.

- Como un control de seguridad adicional desde el lado de la aplicación se puede considerar la adición de envoltorios de seguridad (wrappers) alrededor de los componentes para desactivar las funcionalidades no utilizadas o asegurar los aspectos débiles o vulnerables de los componente.

Capítulo III: Investigación de campo

3.1 Metodología de la investigación

Para poder realizar un diagnóstico de la seguridad de los sitios y aplicaciones web de El Salvador utilizamos la metodología de observación directa al completar un barrido sobre todos los sitios del dominio .sv, que es el asignado a nuestro país.

No todos los sitios web creados en El Salvador, o que tienen su mayor mercado en el país utilizan dominios .sv, ya que es muy popular el usar dominios .com o .org, pero es aceptable considerar que todos los dominios .sv están asociados fuertemente al mercado salvadoreño, y tienen relación con empresas o personas nacionales.

Entre los problemas de seguridad potenciales que ya se han descrito, el uso de componentes con vulnerabilidades conocidas es el más fácil de detectar usando las técnicas correctas, por lo tanto, la investigación se centró en los sitios que están expuestos a este tipo de fallas.

A continuación se describe el proceso de recolección de datos y análisis que se usó para determinar cuáles de los sitios web son potencialmente vulnerables.

3.1.1 Administradores de dominios ".sv":

A nivel mundial está definido el responsable de administrar los identificadores de dominio para cada país, para el caso de El Salvador, la organización encargada de esta tarea es SVNET.

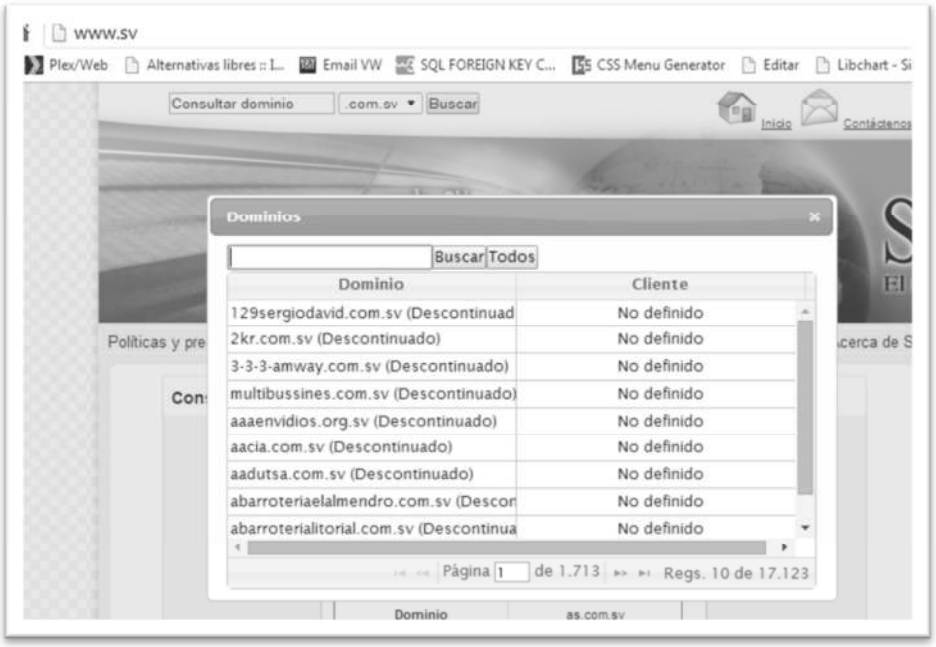
SVNET se encarga de administrar y resolver todos los nombres de dominio que terminen en ".sv", lo que implica que al requerir una dirección de internet que sea su terminación ".sv", hay que acudir a SVNET para que lo agregue a sus servidores "DNS" y los resuelva a una dirección IP publica, ya sea una dirección IP asignada a El Salvador, o que sea una IP de un servidor ubicado en el extranjero.

Para nuestra investigación descargamos todos los nombres de dominio ".sv" del sitio de www.svnet.org.sv (cabe mencionar que el poder descargar todo el directorio de los dominios ".sv" es una vulnerabilidad de este sitio), para posterior mente resolver la IP del servidor que aloja cada sitio web con terminación ".sv". A partir de esta información desarrollaremos nuestra investigación en la que haremos un diagnóstico general sobre las posibles vulnerabilidades que pueden estar teniendo los propietarios de sitios de El Salvador.

3.1.2 Obteniendo el directorio ".sv":

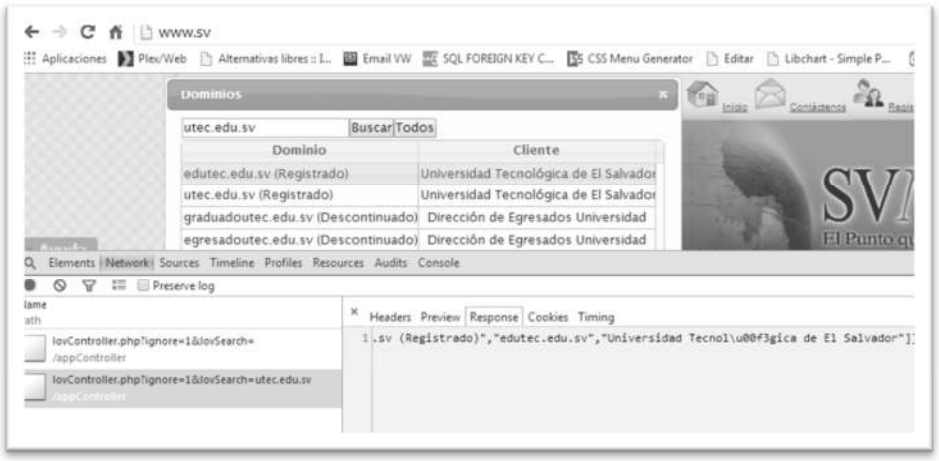
Como ya lo mencionamos en el capítulo anterior existen diversas vulnerabilidades que pueden poner en riesgo nuestra información. El caso del sitio de SVNET no es la excepción: gracias a una vulnerabilidad de como muestran la información, tuvimos acceso inyectar un poco de código JavaScript que nos permitió obtener más información de las que ellos pretenden proporcionarle a sus posibles clientes.

Como primer paso ejecutamos la búsqueda de información de dominios, disponible en la misma página como se muestra a continuación:



Desde este punto ya tenemos acceso a parte de la información que requerimos para nuestra investigación, pero como se puede visualizar únicamente podemos ver una lista de pocos clientes a la vez, y nosotros requerimos contar con el listado completo de los sitios y su estado.

Para lograr dicho objetivo proseguimos identificaremos que tipo de petición se hace cada vez que ejecutamos la búsqueda.



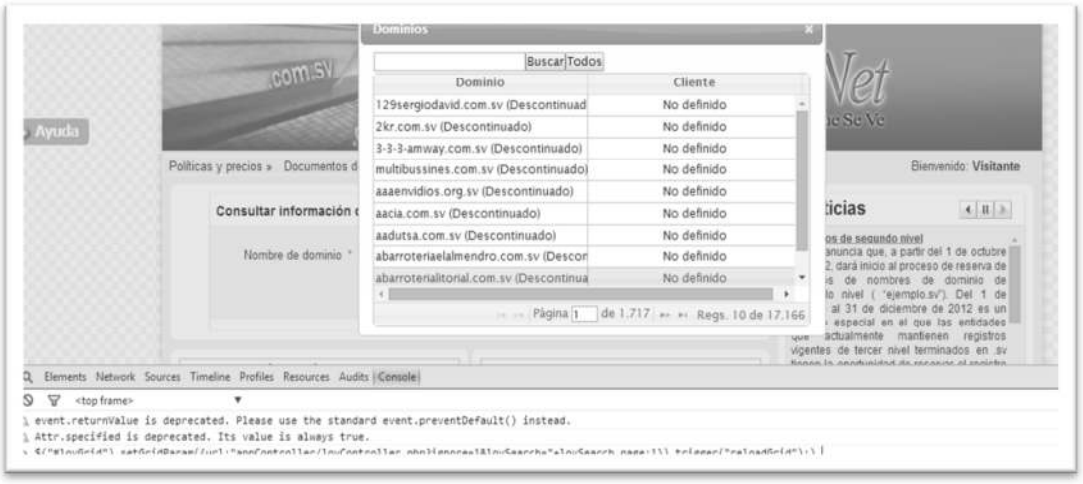
Como se puede observar en la imagen anterior gracias a una útil herramienta que nos proporciona Google Chrome, identificamos que esta búsqueda hace un llamado al controlador de PHP llamado “lovController.php” y este, en su respuesta devuelve una cadena de texto en formato JSON, que fácilmente podemos utilizar posteriormente.

Al investigar un poco nos podemos dar cuenta que esta página utiliza para visualizar la información solicitada un “grid”, este en particular es uno que

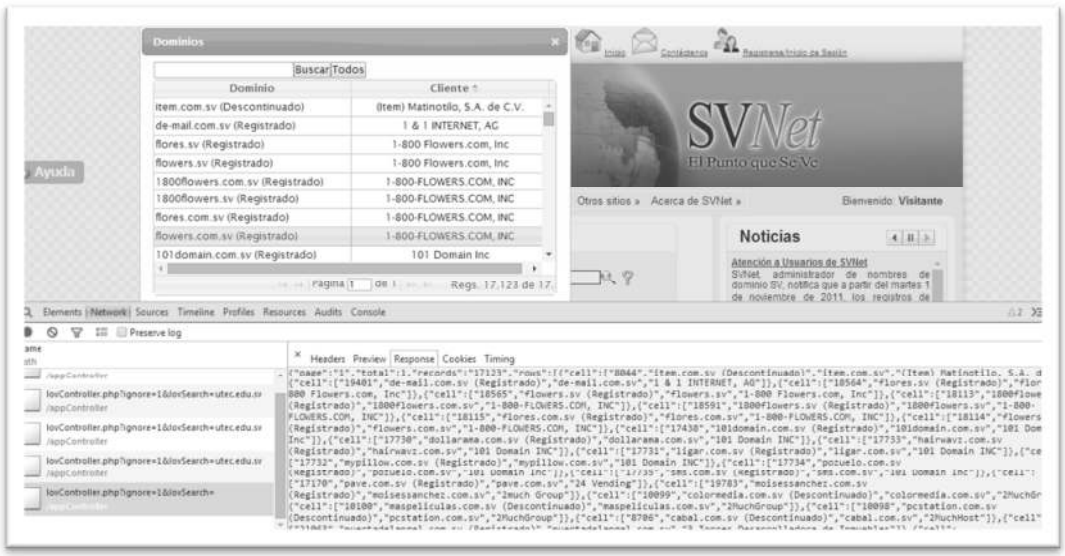
podemos bajar libre mente de internet (“http://www.trirand.com/blog/”) llamado “JQGRID”, a partir de esta información revisamos los archivos JavaScript que utiliza la página, encontrando la línea de código donde identifican dicho objeto como “lovGrid”, asignándole la dirección del controlador y algunos parámetros adicionales.

A partir de este punto, en el que ya se identificó la forma en como recuperan la información, la forma en que la información es mostrada, podemos proceder a inyectar desde la consola (que es otra herramienta que podemos encontrar en Google Chrome o en la mayoría de navegadores) la siguiente línea de código JavaScript.

$("#lovGrid").setGridParam({url:"appController/lovController.php?ignore=1&lovSearch="+lovSearch,page:1}).trigger("reloadGrid");)



Como resultado a este comando tenemos que el controlador nos respondió con el listado completo que se tiene registrada en la base de datos de SVNET, y a su vez, el grid ya no muestra las mismas filas limitadas sino más bien el contenido completo que está retornando el controlador (ver imagen que se muestra a continuacion).



Con la informacion que fue retornada de SVNET procederemos a hacer un escaneo de todas las direcciones de dominio que obtuvimos, identificando alguna infomacion que puede ser de mucho valor en el momento que un atacante identifique posibles victimas. Debido a que muchos sitios están usando aún tecnologías que tienen debilidades identificadas, o están desarrollados en lenguajes de programacion que son reconocidos por ser vulnerables, para lograr identificar algunas de estas variables procederemos de la siguiente forma.

3.1.3 Nslookup dominios ".sv":

Ya con la lista de todos los dominios que nos interesan hay que recuperar la dirección IP del host en la que resuelve el dominio (la dirección IP del hosting); esto nos servirá para poder hacer una distribución geográfica sobre la preferencia de las empresas y técnicos salvadoreños para alojar los sitios, y posteriormente poder hacer un mapa de riesgo de los servidores hosting salvadoreños contra los servidores en el extranjero.

Para alcanzar este objetivo se ha desarrollado una aplicación utilizando "Visual C# de la distribución de Visual Studio 2012", a continuación se describe las clases del Framework 4.0 que se han utilizado y la forma en que fueron implementadas.

3.1.4 Clase "System.Net.Dns":

Es una clase estática que nos permite recuperar información básica de acerca de un host, por lo que fue utilizada para resolver los nombres de los dominios y así obtener la dirección IP que posteriormente utilizaremos.

3.1.5 Clase “System.Net.HttpWebRequest”:

Clase utilizada para hacer una petición web desde el sistema, la cual con la respuesta nos enviara los encabezados que utilizaremos para nuestra investigación.

3.1.6 Geolocalización:

Las direcciones IP están asignadas por cada país, al poseer ya los nombres de dominios y las direcciones IP correspondientes para cada host de las direcciones “.sv” vamos a recuperar la ubicación geográfica para cada dirección IP.

Existen diferentes empresas en internet que nos pueden indicar las coordenadas de una dirección IP, en este caso utilizamos http://www.ipinfodb.com/ que posterior a un registro básico, nos permite consultar las coordenadas de la ubicación para cada dirección IP. Para implementar esta API, nos valimos de la clase “System.Xml.Linq.XDocument” la cual cuenta con un método llamado “Load” el cual recibe como parámetro una ruta, en este caso la dirección URL del API que utilizamos indicando la IP y un token de registro, con esto ya tendremos la información de las coordenadas geográficas, así como el nombre y código del país y ciudad a la cual pertenece dicha dirección IP.

3.1.7 Procesando información de dominios.

La aplicación desarrollada para poder extraer la información de todos los dominios carga la cadena en formato JSON que obtuvimos en los primeros pasos, y luego implementa el Nslookup que se explicó previamente, obteniendo información de la dirección del servidor que contiene el sitio para cada dominio especifico.

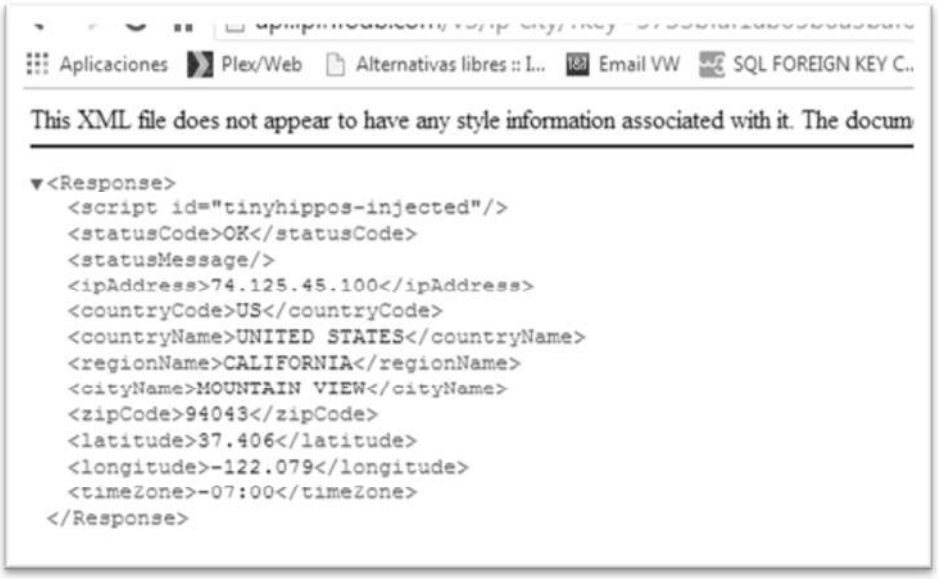
Luego de cargar la cadena JSON que obtuvimos de SVNET, el sistema desarrollado se encarga de escanear uno a uno cada nombre de dominio recuperando la dirección IP de cada host, con una cadena de texto similar a la siguiente.

> *"Estado:OK|Dominio: law.sv|IPs:184.168.224.177|Alias:|MSG:|Extra: law.sv (Registrado)"*

Este tipo de respuesta indica un resultado satisfactorio, y, cuando no es posible obtener la dirección IP, la respuesta es como se indica a continuación:

> *"Estado:ERROR|Dominio:1.com.sv|IPs:|Alias:|MSG:System.Net.Sockets.SocketException (0x80004005): El nombre solicitado es válido pero no se encontraron datos del tipo solicitado"*

A su vez el sistema también recupera la información de Geolocalización de cada dirección IP utilizando el API que se mencionó anterior mente; dicha API nos retorna la información en formato XML, de forma similar a como se presenta en el siguiente ejemplo:



La información resultante de todo este proceso se almacena en una base de datos construida en SQL Server 2008 con el fin poder procesar los datos de una forma más ágil y efectiva posteriormente.

Ha continuación se muestra la interfaz gráfica de la aplicación desarrollada con el fin de hacer los procesos que se explican anterior mente.

3.1.8 Obtención de información de encabezados.

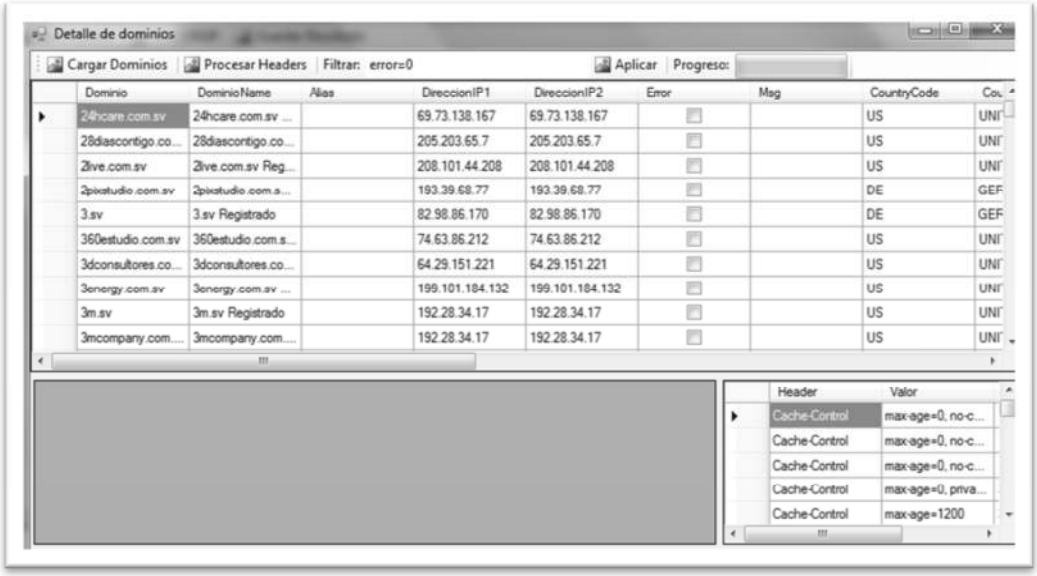
Todas las transacciones generadas en http, ya sea de petición o respuesta incluyen en ellas cierta información llamada “Headers” o encabezados, estos encabezados en muchas ocasiones contienen información muy importante, que podría poner en evidencia que un sitio es vulnerable a ciertos tipos de ataques a través de la web, una versión de servidor desactualizada podría poner en evidencia que el sitio es propenso a ciertos tipos de ataques, o también un lenguaje de programación que es propenso o vulnerable a determinados tipos de ataques.

Partiendo de esta premisa procederemos a recuperar los encabezados de respuesta para todos los sitios a los que les obtuvimos una dirección IP. Para

poder obtener la información de todo el universo de sitios, construimos en nuestra aplicación un módulo que se encarga de esta tarea, en los pasos que se detallan a continuación.

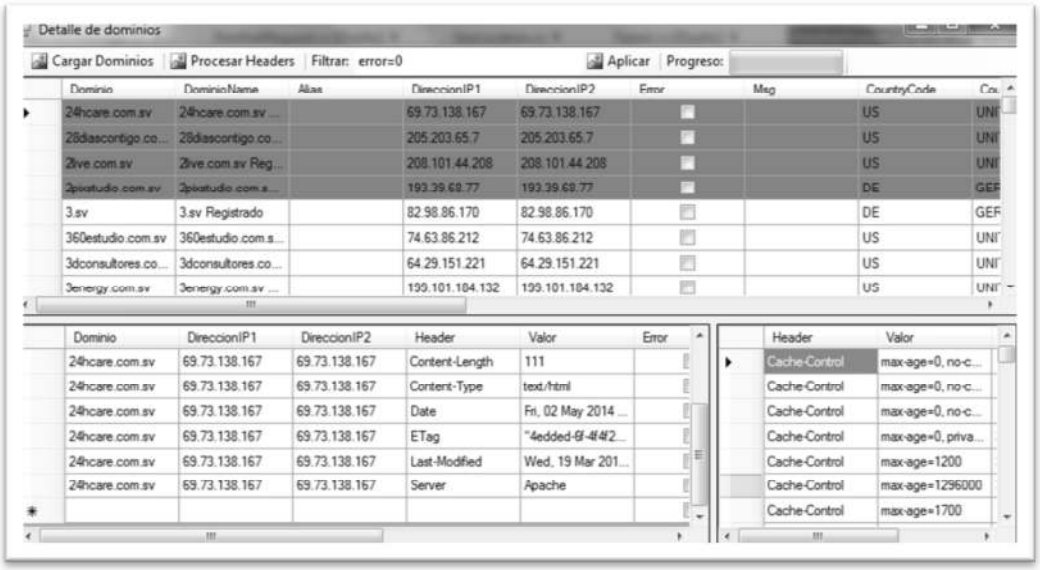
- Recuperamos la información de todos los dominios de los que no obtuvimos ningún error al momento de recuperar la dirección IP.

  Como se muestra en la imagen a continuación, vemos todos los dominios de los que obtuvimos una respuesta satisfactoria con la dirección IP del Host que contiene el sitio web.



- Hace una petición Web utilizando la clase “System.Net.WebRequest” con lo que obtenemos una respuesta que es una instancia de “System.Net.HttpWebResponse”, en este objeto de respuesta está contenida la información de todos los encabezados que retorna el sitio.

Como se muestra en la imagen a continuación, vemos que el sistema procesa uno a uno los nombres de dominio y abajo vemos el resultado de los encabezados de respuesta que se obtuvieron, en los que nos pueden indicar desde nombre de servidor, hasta lenguaje de programación que fue utilizado para la programación del sitio.



3.2 Población y muestra

De acuerdo con la información que se obtuvo de la página de SVNET, ellos reportan un total de 16,741 dominios registrados, entre activos, inactivos y reservados.

De esta población utilizaremos únicamente los dominios activos, más bien los dominios de los que obtuvimos una dirección IP, que son un total de 4,920 representando un 29.4% del total de dominios que reporta SVNET en su página web.

Para delimitar el universo de datos sobre los cuales se centrará nuestra investigación, seleccionamos por medio de la ubicación geográfica de cada IP correspondiente a cada host, los once países que reportan mayor número de hospedajes de sitios web.

El cuadro siguiente nos indica la cantidad de servidores que hospedan sitios .sv y que se tomaran en cuenta en la investigación sub dividido por país.

País	**No. Sitios**
Estados Unidos	3373
El Salvador	721
Alemania	268
Reino Unido	131
Canadá	67
Francia	50
Argentina	39
España	37
Australia	33
Guatemala	31
México	19
Total	**4769**

Esto nos deja con 4,769 dominios los cuales representan un 97% del total de dominios que respondieron con una dirección IP.

Tomando como muestra este total de 4,769 dominios que respondieron con una dirección IP, y que están dentro de los once países más representativos del total de respuestas satisfactorias, presentaremos el resultado de nuestra investigación.

3.3 Análisis e interpretación de los datos

Luego de obtener y almacenar los datos en nuestra base de datos utilizando el software que desarrollamos, procedemos a utilizar las muchas ventajas que nos proporciona el lenguaje Transact-SQL para extraer los datos que nos interesan para el propósito de esta investigación.

Con ese fin utilizamos los valores de los encabezados de solicitud que contenían valores que nos interesan, entre los que podemos mencionar: Server, ServerName, X-AspNet-Version, X-Powered-By.

3.3.1 Guía de observación

La información recopilada se utilizó para responder a las preguntas de la guía que se presenta a continuación.

Las primeras 4 preguntas tienen como propósito identificar cuáles son los servidores que proporcionan información que podría ser usada por un atacante para atacar al servidor.

El segundo grupo de 6 preguntas analiza las tendencias de uso de tecnologías web en base a la información recopilada, y agrupa los resultados por el país donde se ubican los servidores que responden a los dominios .sv, según la técnica de geolocalización descrita anteriormente (limitada a los 11 países más significativos).

3.3.2 Preguntas de análisis

1 ¿Cuántos de los dominios .sv están activos?

2 ¿En qué países se ubican los servidores a los cuales resuelven los dominios .sv activos?

3 ¿Cuáles de los servidores identificados responden a peticiones HTTP enviando encabezados?

4 ¿Cuáles de los servidores que envían encabezados exponen en ellos detalles de posibles vulnerabilidades?

5 ¿Qué tecnología de servidor web usan los servidores de dominios .sv dependiendo del país donde se ubican?

6 ¿Cuál es la versión de servidor Apache más utilizada?

7 ¿Cuál es la versión de IIS más utilizada?

8 ¿Cuáles son los lenguajes de programación más utilizados por país donde se aloja el servidor?

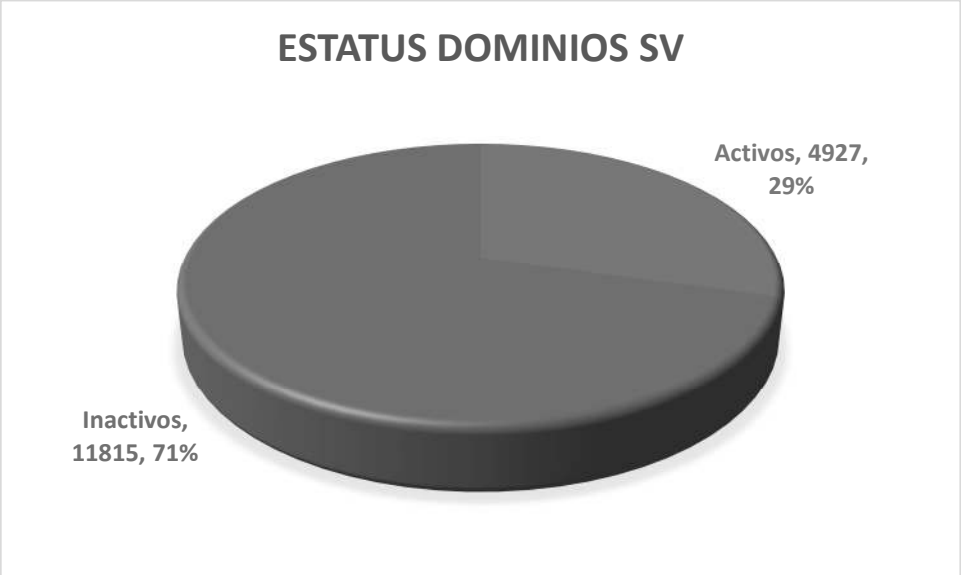
9 ¿Cuál es la versión de PHP más utilizada?

10 ¿Cuál es la versión de ASP más utilizada?

3.4 Presentación de los resultados

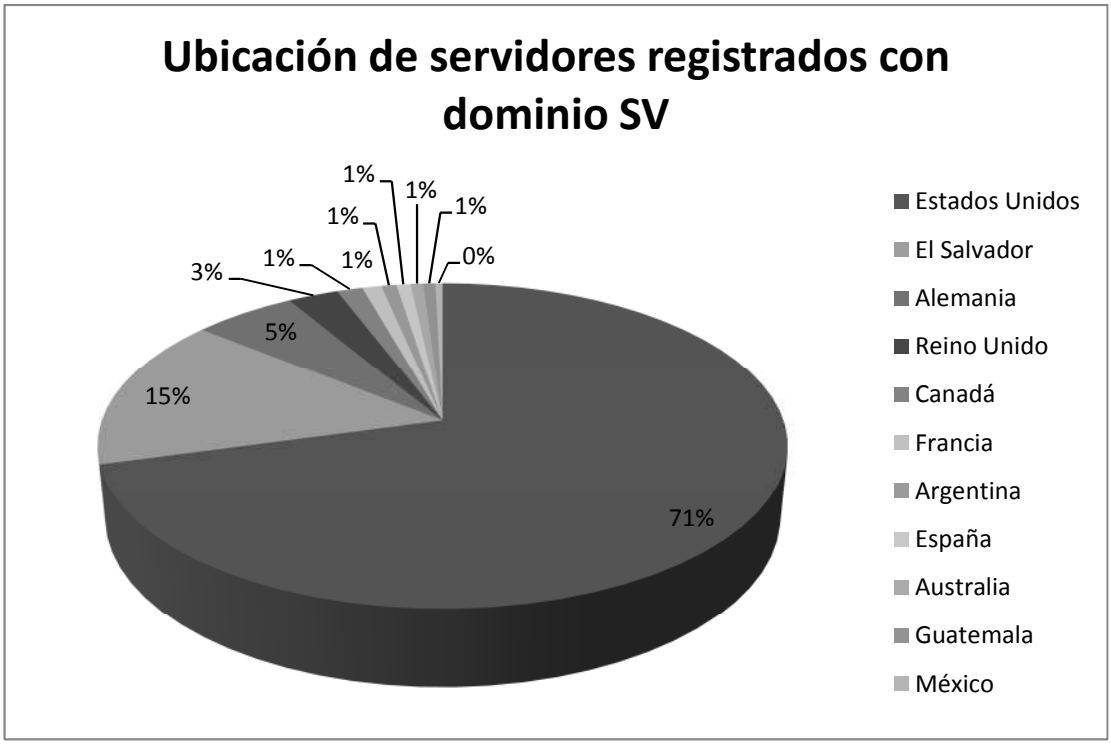
A continuación se presentan los resultados obtenidos de la investigación de campo, en los que podremos observar una imagen clara de la tendencia para los dominios en El Salvador.

3.4.1 ¿Cuántos de los dominios .sv están activos?



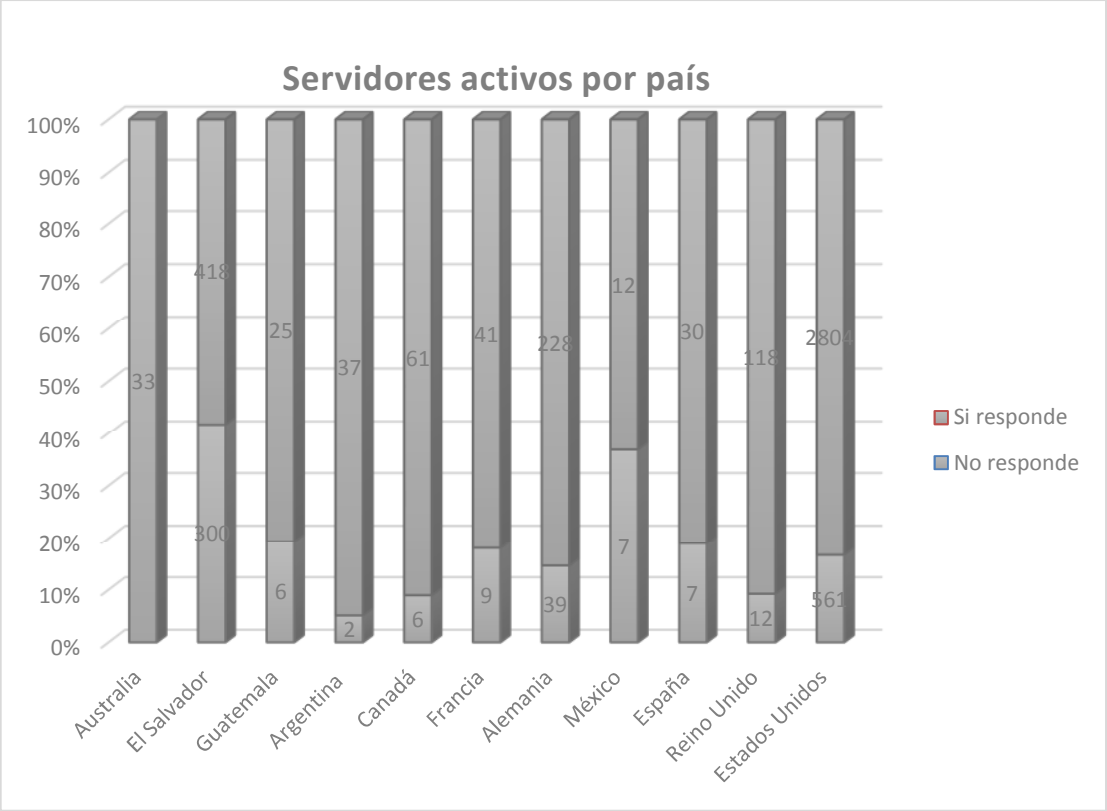
Con los datos obtenidos al mes de febrero de 2014 existen un total de 16,742 dominios registrados en SVNET de los cuales únicamente 4,927 es decir el 29% están activos, y un 71% de los dominios sigue estando reservado por ellos pero no están en uso.

3.4.2 ¿En qué países se ubican los servidores a los cuales resuelven los dominios .sv activos?



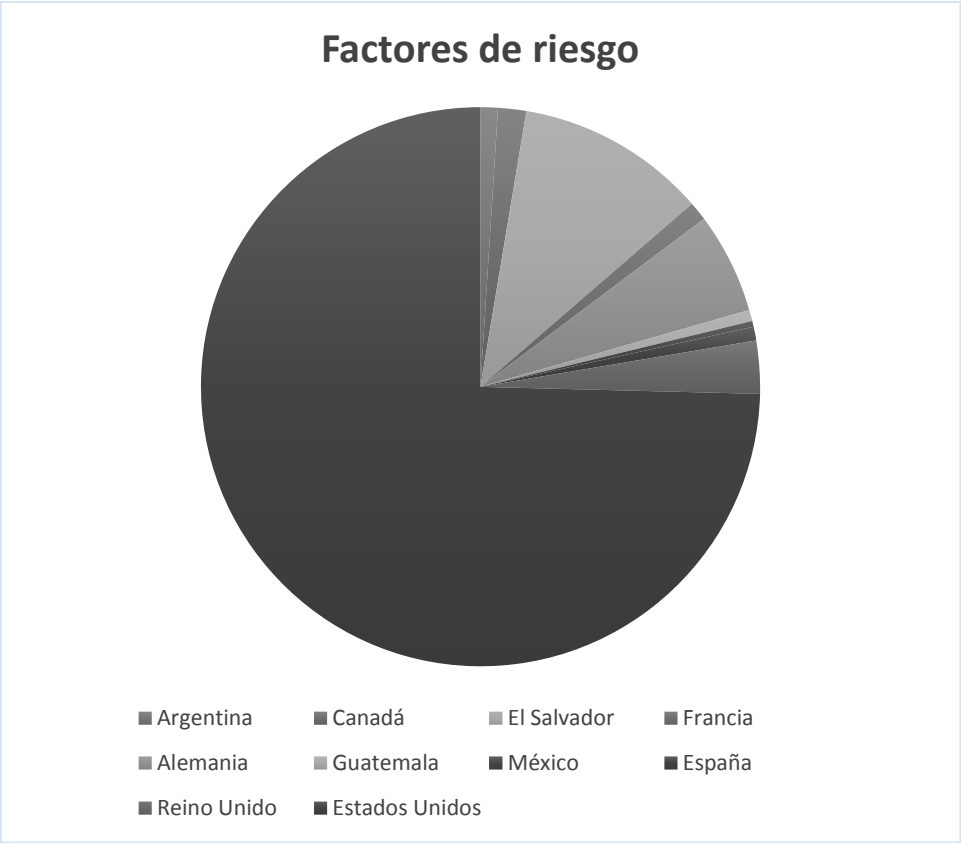
Como su título lo indica este gráfico nos ilustra cómo están distribuidos geográficamente los servidores host para los dominios en El Salvador. Estados Unidos representa el mayor proveedor de este servicio, seguido de El Salvador como los más representativos, aunque cabe recalcar que países como Alemania y Reino Unido participan como proveedores importantes de este servicio para El Salvador también.

3.4.3 ¿Cuáles de los servidores identificados responden a peticiones HTTP enviando encabezados?



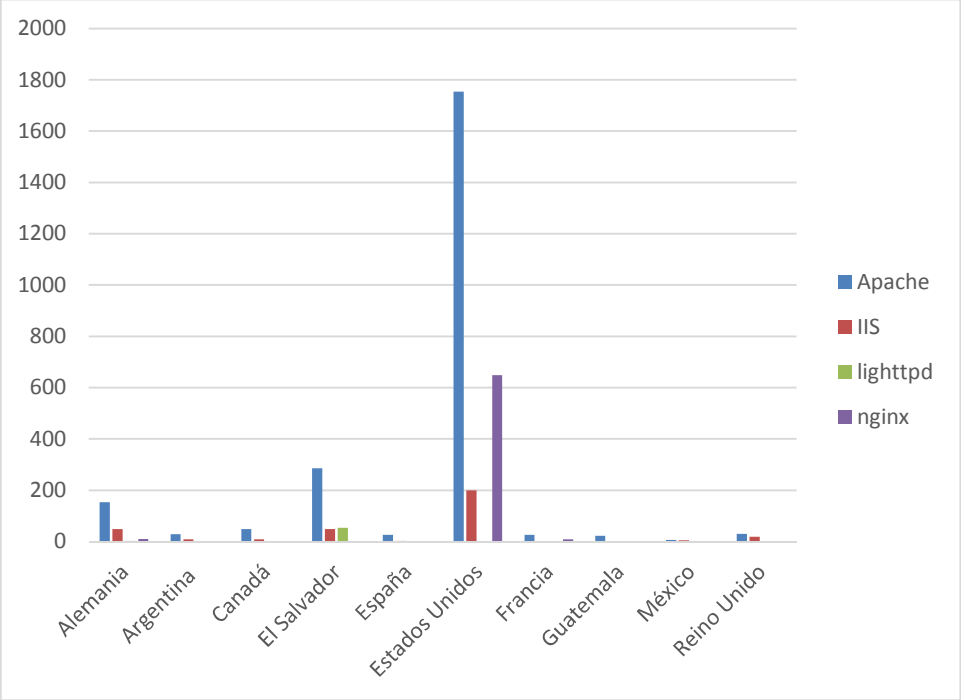
Este gráfico nos indica qué sitios, de los ya filtrados nos respondieron con algún tipo de encabezado, cabe mencionar que no todos los encabezados representan información necesariamente útil para ser considerada como posible vulnerabilidad.

3.4.4 ¿Cuáles de los servidores que envían encabezados exponen en ellos detalles de posibles vulnerabilidades?



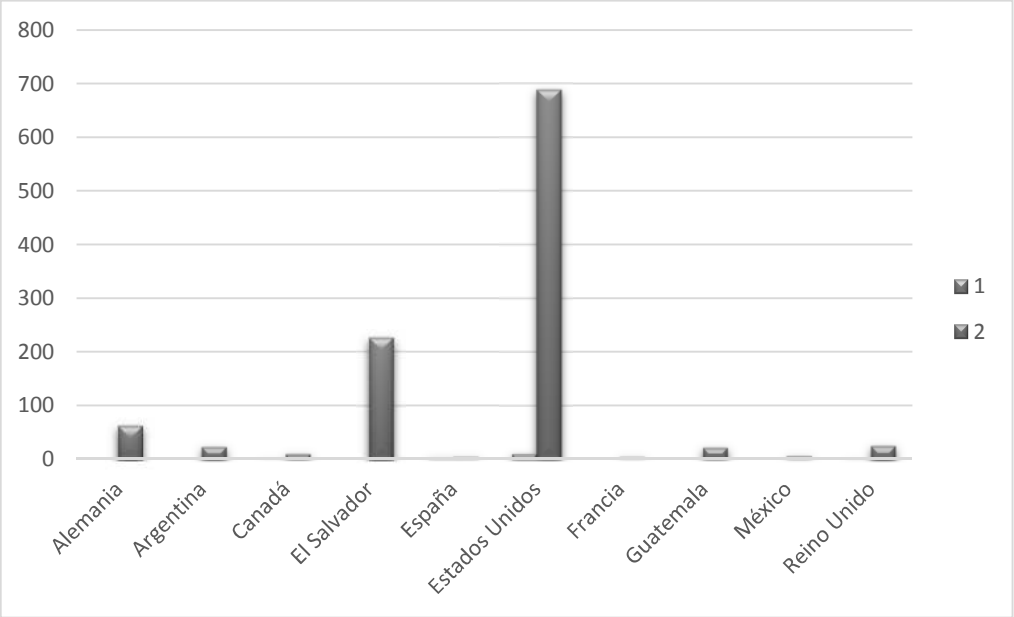
Este gráfico nos indica en qué cantidad podemos obtener información que puede ser utilizada en forma maliciosa, como se observa en el gráfico, de la misma forma que Estados Unidos es el país más representativo en el alojamiento de sitios es también el país que nos entrega encabezados con información que podemos utilizar para atacar.

3.4.5 ¿Qué tecnología de servidor web usan los servidores de dominios .sv dependiendo del país donde se ubican?



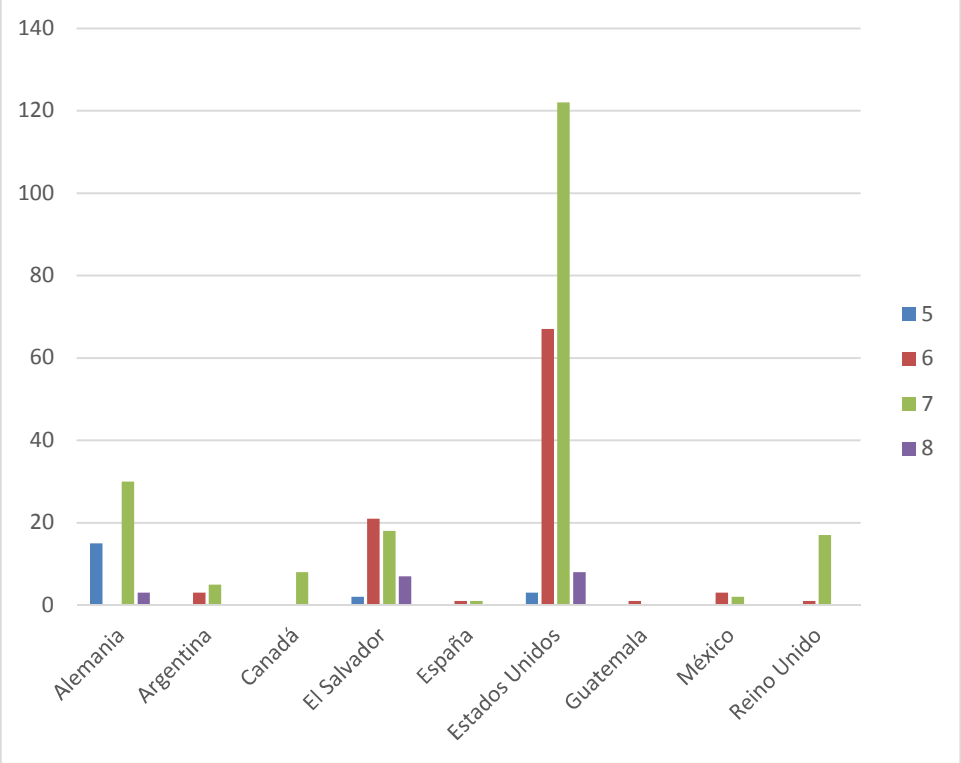
El gráfico anterior muestra los tipos de servidor más utilizados, por cada país, como lo indica el gráfico, Apache es el servidor más utilizado en Estados Unidos, lo que nos indica una tendencia de predilección para los sistemas de no pago de licenciamiento.

3.4.6 ¿Cuál es la versión de servidor Apache más utilizada?



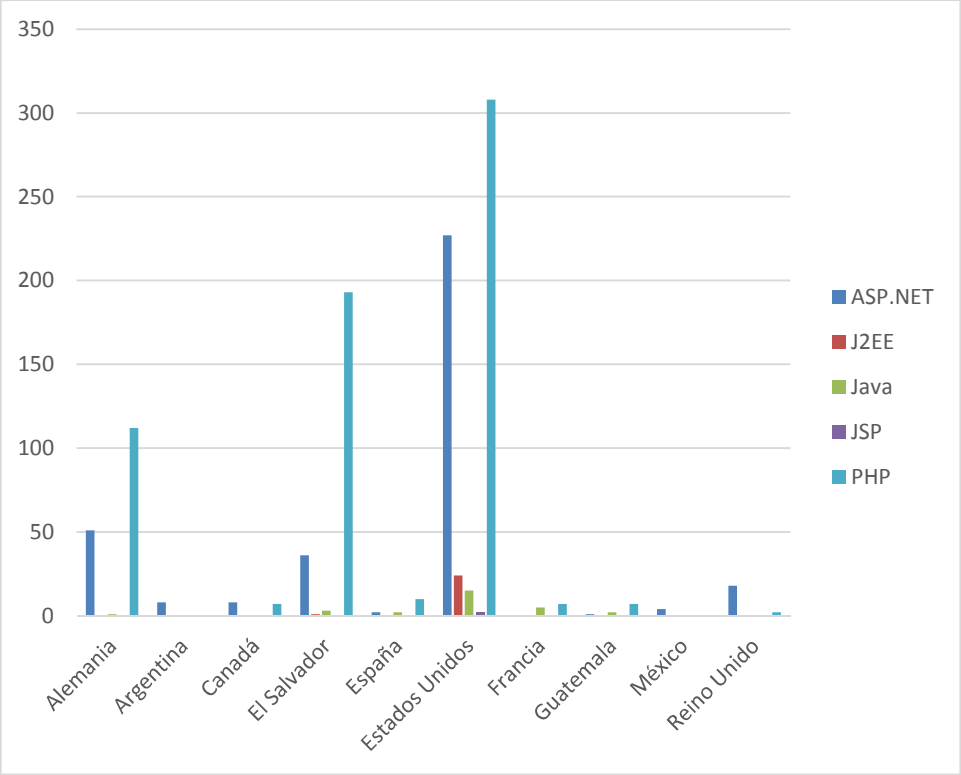
En el gráfico anterior podemos visualizar la versión de servidores Apache que se utilizan en los diferentes países que lo usan, tal cual lo indica el gráfico, la versión dos es la más usada, aunque podemos observar que aún se utiliza la versión uno en los diferentes países, siendo este un factor de riesgo en relación a los que usan la versión dos.

3.4.7 ¿Cuál es la versión de IIS más utilizada?



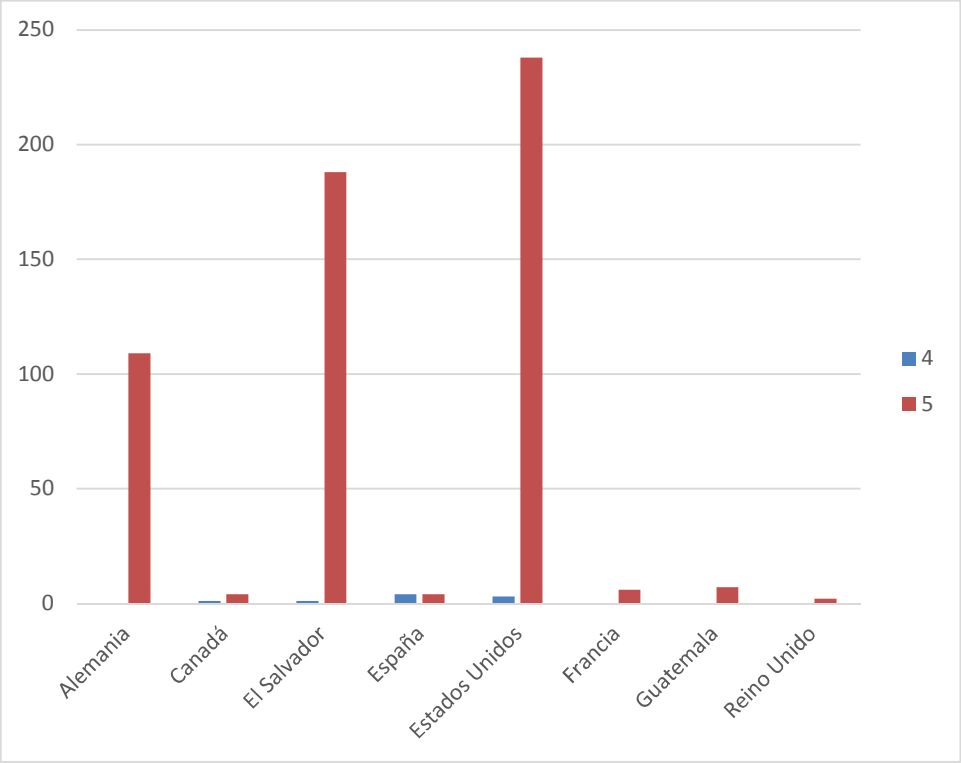
En este gráfico podemos observar la versión de servidor IIS más utilizada en los diferentes países, vemos como la versión ocho siendo la más reciente, aún no es la más utilizada, y como factor de riesgo vemos que la versión seis y cinco sigue siendo utilizada por algunos países, esto implica un factor de riesgo, pues cuando la versión es más antigua, está sujeta a fallas ya conocidas, muchas de las que pueden ser muy críticas.

3.4.8 ¿Cuáles son los lenguajes de programación más utilizados por país donde se aloja el servidor?



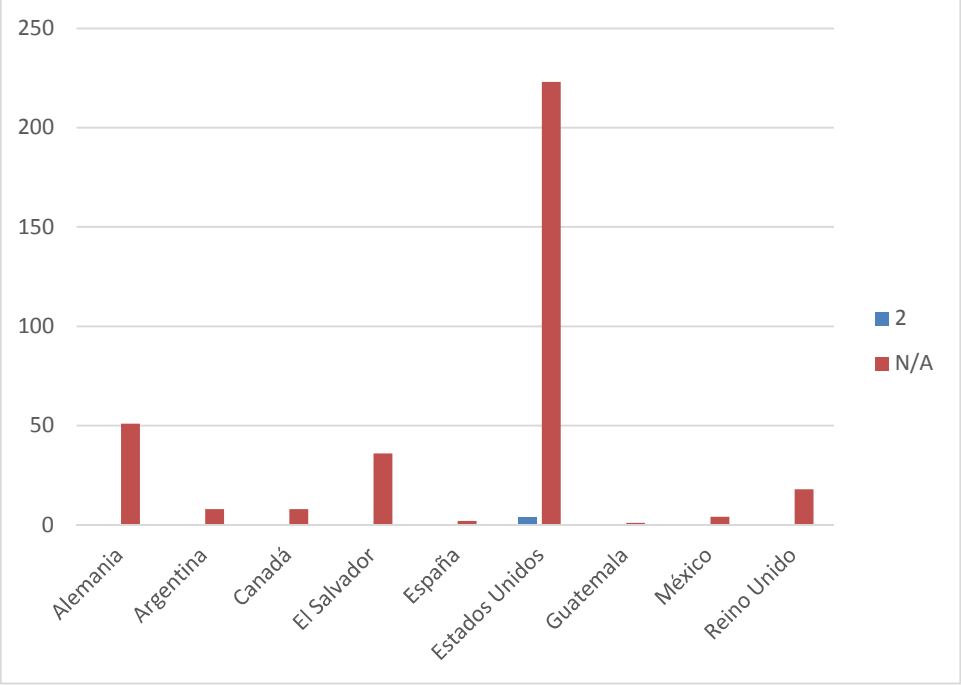
El gráfico nos indica que tipo de lenguaje de programación es más utilizado, para servidores de dominios .sv; claramente vemos que PHP ocupa el primer lugar en Estados Unidos, pero es seguido por ASP.NET. Esta información es de mucha importancia, pues determinando el lenguaje y la versión de dichos lenguajes, un atacante identifica a qué tipo de vulnerabilidades puede estar sujeto un sitio determinado.

3.4.9 ¿Cuál es la versión de PHP más utilizada?



Entre los servidores hospedados en los países, que respondieron con la versión de lenguaje PHP que utilizan, podemos observar que en su mayoría trabajan con la versión cinco, sin embargo, vemos como en El Salvador, existen aún páginas que utilizan la versión 4, así como también sucede en países que son referentes tales como Estados Unidos o España.

3.4.10 ¿Cuál es la versión de ASP más utilizada?



La información planteada en este gráfico es sumamente interesante, ya que observamos cómo aunque los sitios retornan qué lenguaje de programación utilizan, estos mismos no retornan la versión del lenguaje utilizada, por lo que para un posible atacante le será un poco más complicado elegir el tipo de ataque adecuado, de todos los sitios que indicaron usar ASP.NET únicamente para Estados Unidos se encontró que cuatro sitios respondieron que utilizaban la versión dos de ASP.NET, esto los podría convertir objetivos de futuros ataques, ya que esta versión es vulnerable a ciertos ataques XSS.

Capítulo IV: Caso de estudio

Las organizaciones deben estar listas para atender los casos de fallas de seguridad en sistemas informáticos con un proceso documentado, que recopile toda la información necesaria y que permita al personal responsable determinar el impacto, la duración, y posibles riesgos generados por la vulnerabilidad detectada.

Los problemas de seguridad pueden ser detectados por auditorías de seguridad de sistemas o de código, pruebas internas, ejecución de herramientas automatizadas de análisis, o por reportes de errores provenientes del exterior de la organización.

Durante el proceso de investigación del presente documento detectamos un sitio web de un banco internacional que opera en El Salvador que sufría de una serie de fallas y vulnerabilidades con riesgos medianos a altos, el nombre de este banco, y del personal que trabajó para resolver los problemas detectados se mantendrá confidencial.

Una parte importante de este caso es el análisis de los procedimientos y directrices del banco para responder a incidentes de seguridad, y la respuesta del equipo de informática, y otras personas involucradas en la investigación,

más un seguimiento del incidente de seguridad de la información notificado. (Hilary Baker, VP for IT, 2009)

4.1.1 Identificación de vulnerabilidades

El día 29 de agosto de 2013 se encontraron una serie de vulnerabiliadades en el sitio de redención de puntos de un banco reconocido en El Salvador:

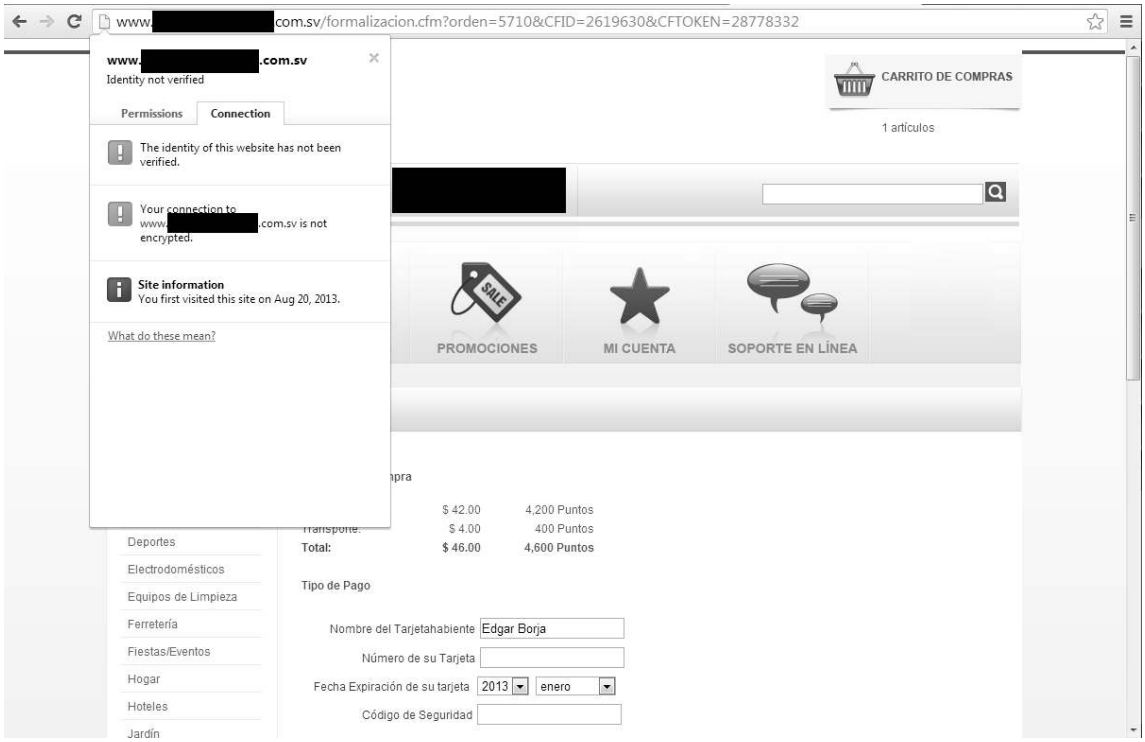


Figura 1. Captura de pantalla del sitio web mostrando vulnerabilidades

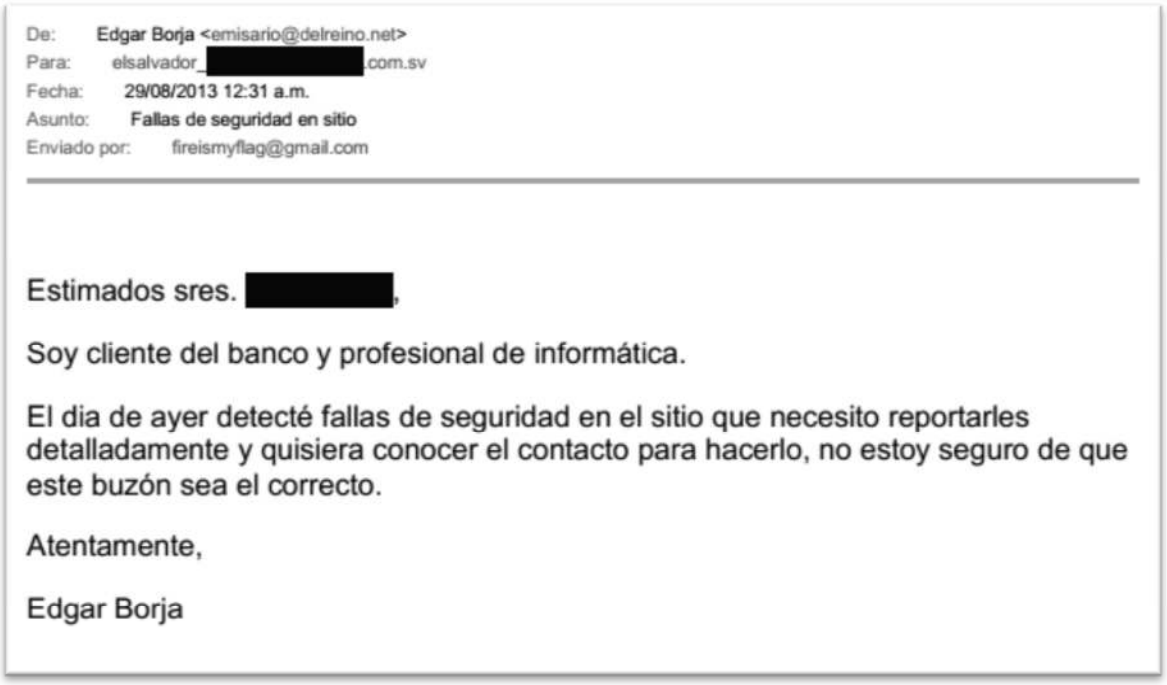
Las vulnerabilidades que se detectaron se listan a continuación:

- Fallas de autenticación y gestión de sesiones: el sitio presenta en la URL fragmentos importantes del proceso de autenticación, como el identificador único de la sesión del usuario. Para volver más crítica esta falla, si se copia la URL y se abre desde cualquier otra computadora, el navegador cargará la sesión abierta del usuario inicial. La URL puede ser descubierta a través de cualquier proxy o en registros del navegador o de la computadora.
- Configuración de seguridad incorrecta: Como se puede observar en la figura 1, el sitio no utiliza un certificado para encriptar la conexión con el navegador, por lo que todas las acciones del usuario son potencialmente vulnerables a ser leídas o registradas en cualquier punto del trayecto entre el navegador y el servidor.
- Exposición de datos sensibles: El sitio solicita al usuario que introduzca su nombre, número de tarjeta de crédito, fecha de expiración y código de seguridad, todos estos datos son luego enviados en una conexión no encriptada, y en base a la falta de estándares que se observa, probablemente almacenado sen una base de datos no encriptada.

El detalle de las implicaciones de estas vulnerabilidades se presentaron en el marco conceptual del presente documento, donde también se incluyeron las recomendaciones para su mitigación.

4.1.2 Reporte de las fallas.

A continuación se presenta la cadena de correos que iniciamos cuando reportamos inicialmente las amenazas que se detectaron:



De: Edgar Borja <emisario@delreino.net>
Para: elsalvador_[redacted].com.sv
Fecha: 29/08/2013 12:31 a.m.
Asunto: Fallas de seguridad en sitio
Enviado por: fireismyflag@gmail.com

Estimados sres. [redacted],

Soy cliente del banco y profesional de informática.

El dia de ayer detecté fallas de seguridad en el sitio que necesito reportarles detalladamente y quisiera conocer el contacto para hacerlo, no estoy seguro de que este buzón sea el correcto.

Atentamente,

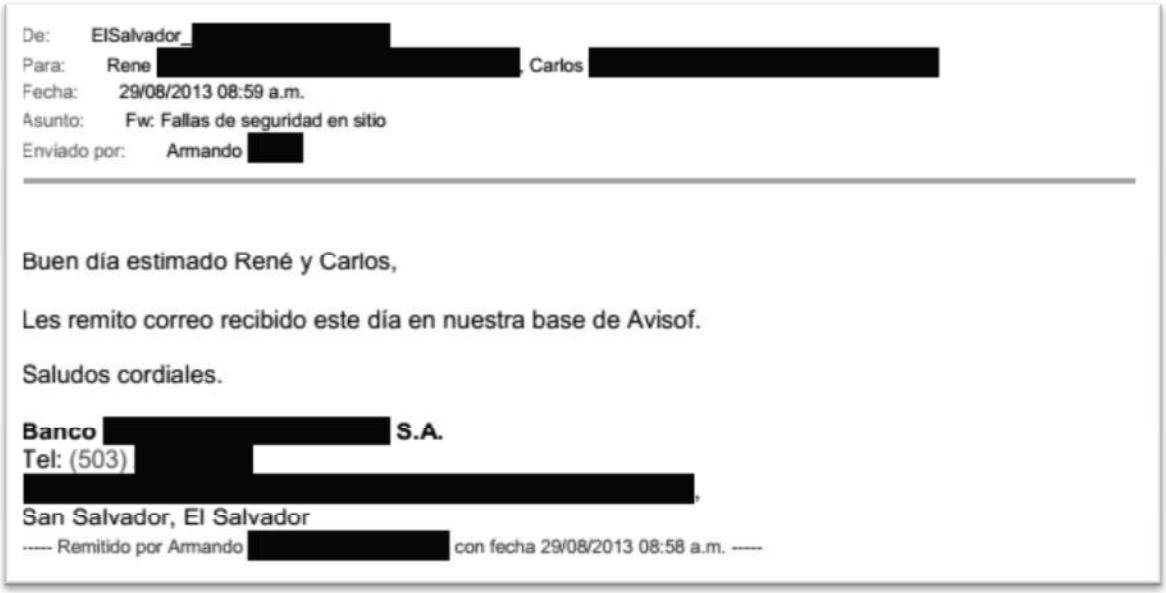
Edgar Borja

Esta primera notificación estaba dirigida a un buzón genérico de atención al cliente, por lo que no contenía detalles de las vulnerabilidades detectadas. En

una organización, el área de Informática o Seguridad de la Información puede recibir la notificación de un incidente de muchas áreas: Atención al cliente, operaciones de red, las divisiones de la organización, y el público

Es responsabilidad del área de Informática el establecer procesos claros para que cuando los incidentes sean reportados a por el público, o detectados por otras divisiones, el escalamiento interno del incidente sea realizado con prontitud y a través de los canales apropiados.

Para este caso, el área de atención al cliente escaló la notificación a los responsables de seguridad informática al inicio del siguiente dia laboral.



De: ElSalvador_[redacted]
Para: Rene [redacted], Carlos [redacted]
Fecha: 29/08/2013 08:59 a.m.
Asunto: Fw: Fallas de seguridad en sitio
Enviado por: Armando [redacted]

Buen día estimado René y Carlos,

Les remito correo recibido este día en nuestra base de Avisof.

Saludos cordiales.

Banco [redacted] **S.A.**
Tel: (503) [redacted]
[redacted],
San Salvador, El Salvador
----- Remitido por Armando [redacted] con fecha 29/08/2013 08:58 a.m. -----

Cuando el reporte del incidentes es recibido se le asignará el nivel severidad, por un miembro de la Oficina de Seguridad de la Información, antes de pasar al próximo paso; para el caso de estudio, en vista de que no proporcionamos toda la información inicialmente (por desconocer los canales apropiados) las personas responsables nos contactaron para obtener más información.

Carlos [redacted] <carlos.[redacted].com.sv> Thu, Aug 29, 2013 at 10:27 PM

To: Edgar Borja <emisario@delreino.net>
Cc: Armando [redacted] <armando.[redacted].com.sv>, Rene [redacted] <rene.[redacted].com.sv>

Estimado Sr. Borja:

Gracias por su retroalimentación. Agradeceré mucho si me indica las fallas encontradas.

Atentamente.

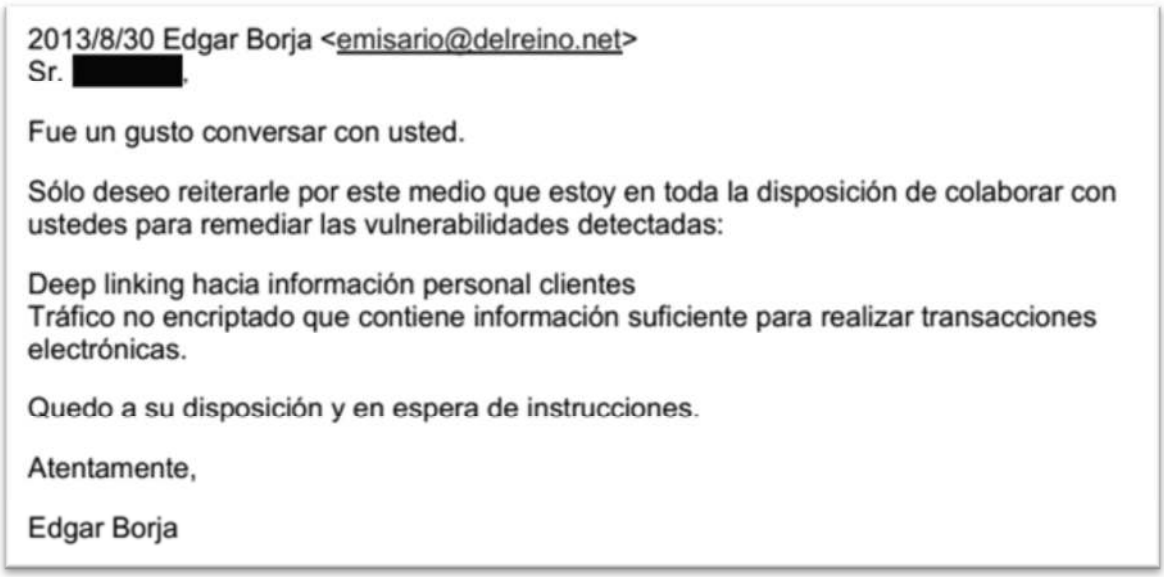
Carlos [redacted]

San Salvador, El Salvador

Phone. (503) [redacted]
Fax. (503) [redacted]
Email. carlos.[redacted].com.sv

Luego de este primer contacto de parte del banco hacia nosotros, nos comunicamos telefónicamente con la persona a cargo del incidente para describirle los problemas detectados, así como exponerle nuestras intenciones de colaborar con ellos para su resolución.

Un incidente que podría tener efectos a largo plazo en los negocios o afectar a los sistemas críticos o tener gran impacto en la organización o podría dañar la reputación de la empresa, por estas razones, son un tema muy sensible y nos optamos por solicitar una reunión personal con los encargados del área, como una forma de demostrar transparencia y el deseo de colaborar.



2013/8/30 Edgar Borja <emisario@delreino.net>
Sr. [redacted],

Fue un gusto conversar con usted.

Sólo deseo reiterarle por este medio que estoy en toda la disposición de colaborar con ustedes para remediar las vulnerabilidades detectadas:

Deep linking hacia información personal clientes
Tráfico no encriptado que contiene información suficiente para realizar transacciones electrónicas.

Quedo a su disposición y en espera de instrucciones.

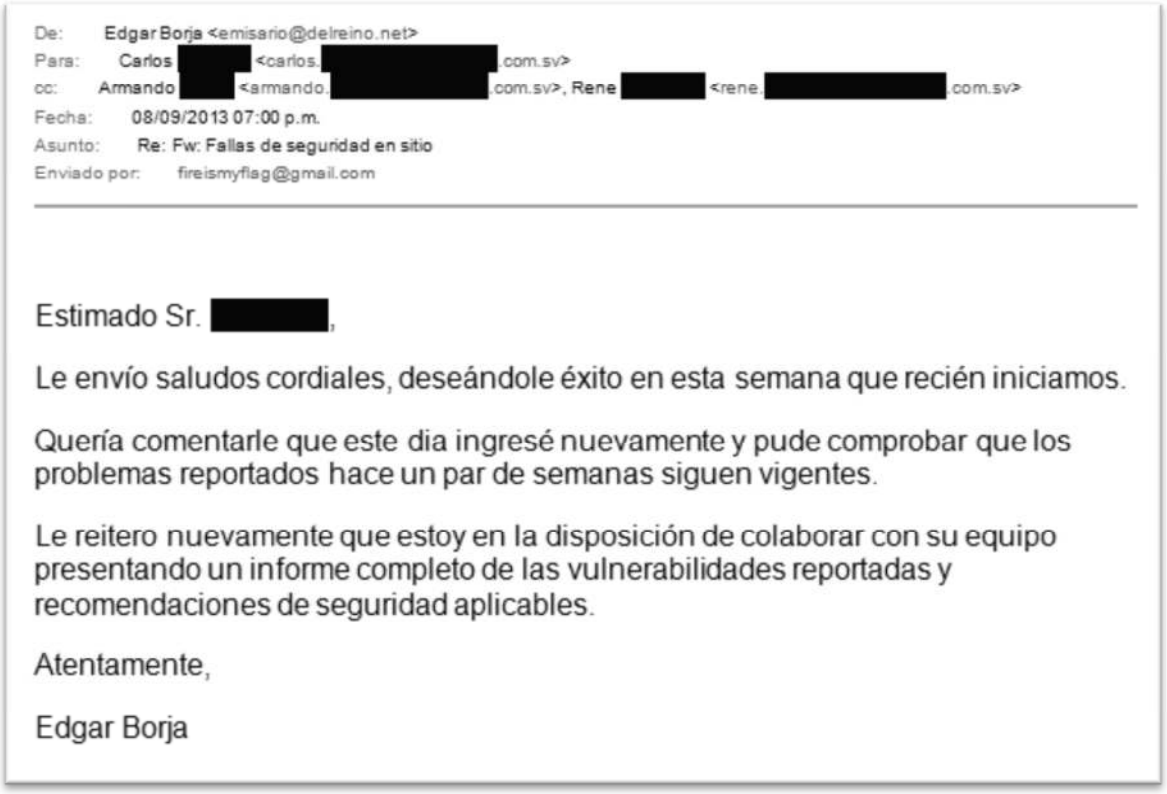
Atentamente,

Edgar Borja

El tiempo de respuesta de los incidentes debe estar definido en un acuerdo de nivel de servicio interno de la organización, y dependerá de la severidad del problema y del nivel de exposición que genera. En el caso del banco, inicialmente experimentaron varios atrasos para iniciar la resolución.

El retraso en la resolución de un incidente de seguridad informática cuando este ya ha sido detectado y reportado incrementa los riesgos a los que la organización puede estar expuesta, en este caso, desde el principio se manejó el incidente con

discreción, pero los incidentes podrían ser reportados en medios públicos como foros, o redes sociales, en todas estas instancias, personas no relacionadas a la organización podrían tener acceso a información clave para reproducir o explotar las vulnerabilidades reportadas, aunque la persona que las haya detectado inicialmente no tenga intenciones de causar daño. En vista de que este problema seguía vigente decidimos insistir.



De: Edgar Borja <emisario@delreino.net>
Para: Carlos [redacted] <carlos.[redacted].com.sv>
cc: Armando [redacted] <armando.[redacted].com.sv>, Rene [redacted] <rene.[redacted].com.sv>
Fecha: 08/09/2013 07:00 p.m.
Asunto: Re: Fw: Fallas de seguridad en sitio
Enviado por: fireismyflag@gmail.com

Estimado Sr. [redacted],

Le envío saludos cordiales, deseándole éxito en esta semana que recién iniciamos.

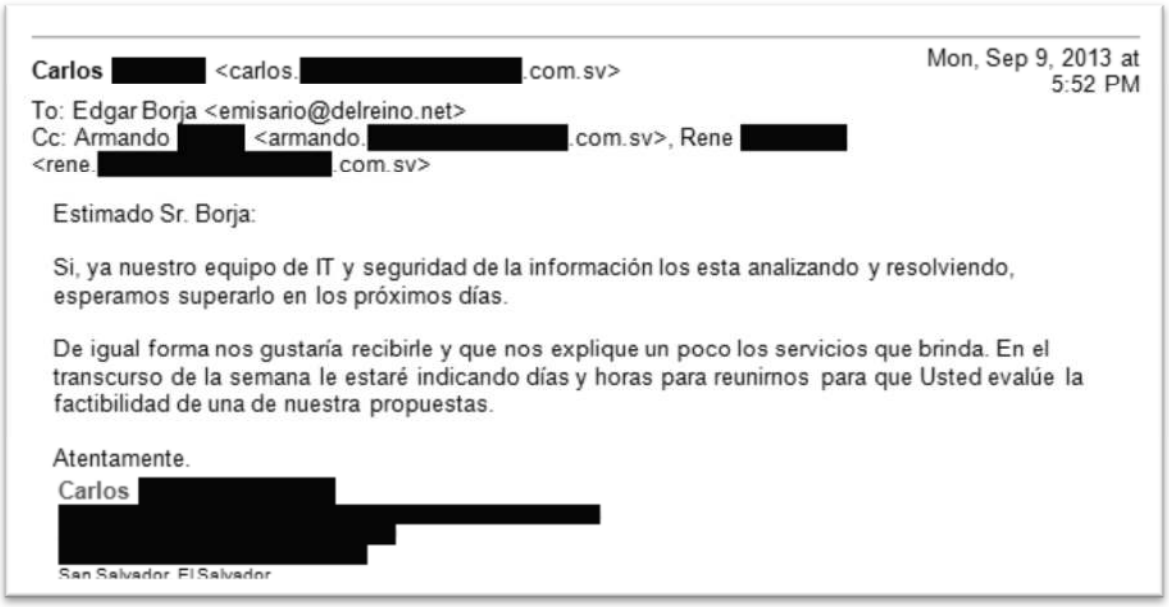
Quería comentarle que este dia ingresé nuevamente y pude comprobar que los problemas reportados hace un par de semanas siguen vigentes.

Le reitero nuevamente que estoy en la disposición de colaborar con su equipo presentando un informe completo de las vulnerabilidades reportadas y recomendaciones de seguridad aplicables.

Atentamente,

Edgar Borja

Luego de este segundo intento fuimos invitados a una reunión.



Carlos [redacted] <carlos.[redacted].com.sv> Mon, Sep 9, 2013 at 5:52 PM

To: Edgar Borja <emisario@delreino.net>
Cc: Armando [redacted] <armando.[redacted].com.sv>, Rene [redacted] <rene.[redacted].com.sv>

Estimado Sr. Borja:

Si, ya nuestro equipo de IT y seguridad de la información los esta analizando y resolviendo, esperamos superarlo en los próximos días.

De igual forma nos gustaría recibirle y que nos explique un poco los servicios que brinda. En el transcurso de la semana le estaré indicando días y horas para reunirnos para que Usted evalúe la factibilidad de una de nuestra propuestas.

Atentamente.
Carlos [redacted]
San Salvador, El Salvador

Respondimos afirmativamente, en vista de que el banco estaba dispuesto a aceptar nuestra ayuda.

Edgar Borja <emisario@delreino.net> Tue, Sep 10, 2013 at 3:16 PM

To: Carlos [redacted] <carlos.[redacted].com.sv>
Cc: Armando [redacted] <armando.[redacted].com.sv>, Rene [redacted] <rene.[redacted].com.sv>

Sr. [redacted],

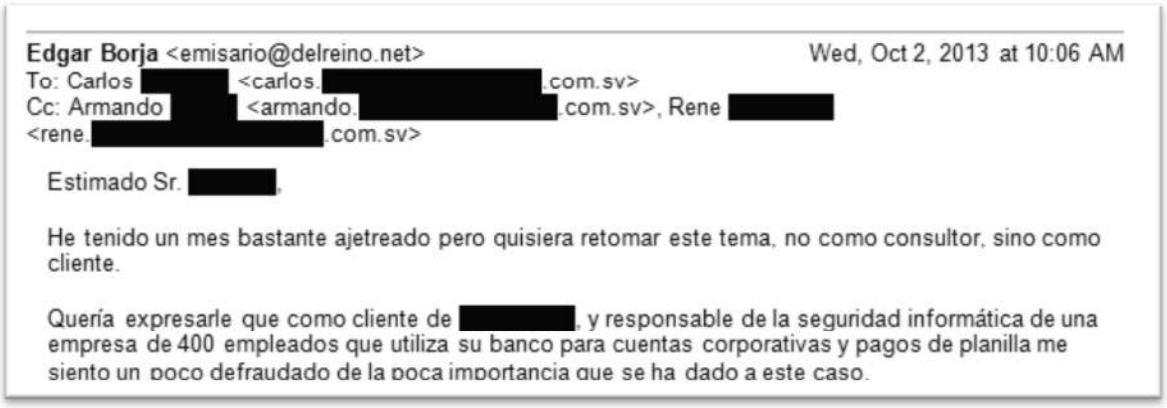
Me alegra la noticia.

Será un gusto visitarle.

Atentamente,

Edgar Borja
[Quoted text hidden]

Sin embargo una vez más experimentamos un sensible retraso para continuar con el proceso. En esta ocasión transcurrieron tres semanas más antes de recibir una respuesta del banco.

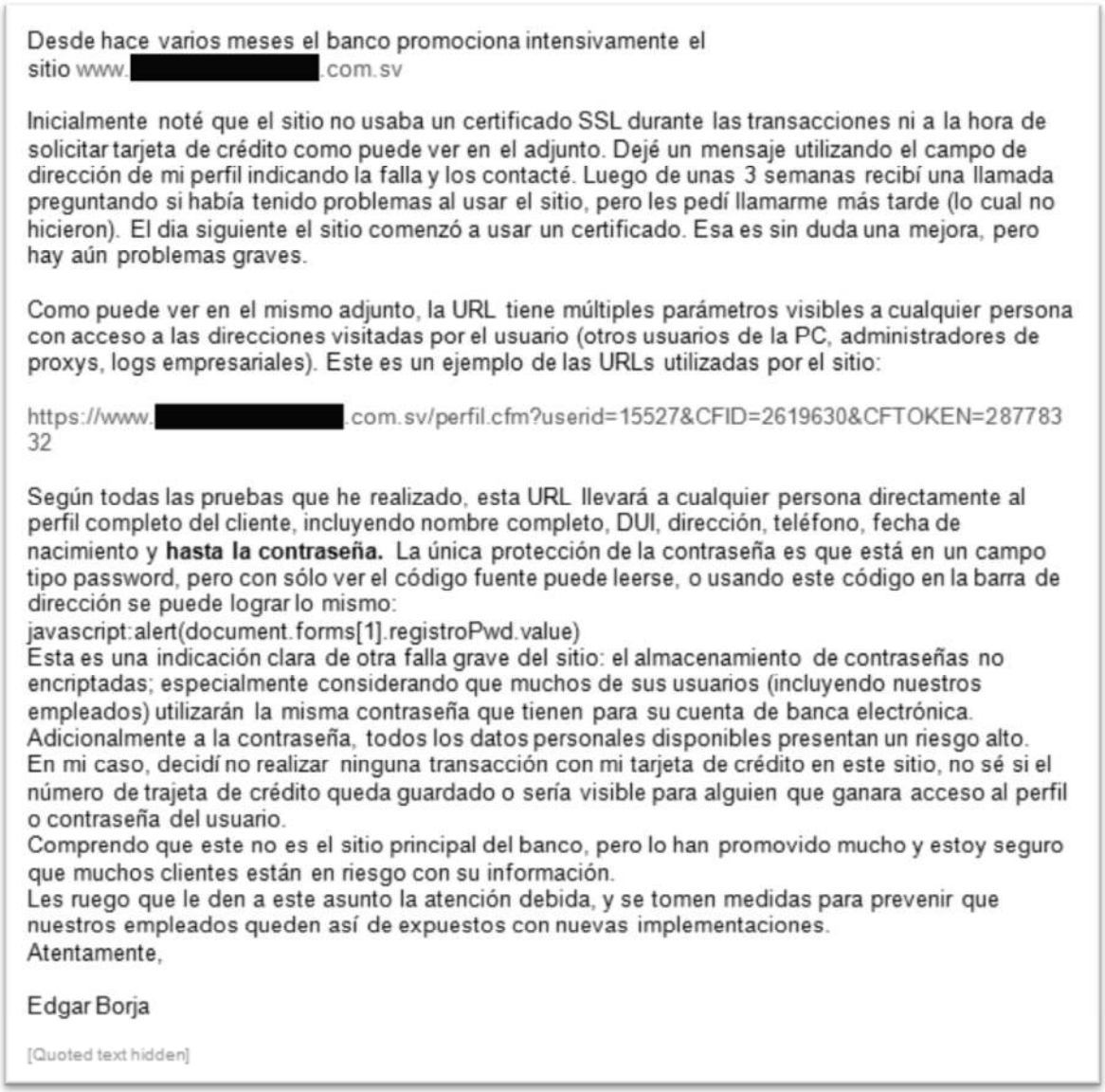


Edgar Borja <emisario@delreino.net> Wed, Oct 2, 2013 at 10:06 AM
To: Carlos [redacted] <carlos.[redacted].com.sv>
Cc: Armando [redacted] <armando.[redacted].com.sv>, Rene [redacted]
<rene.[redacted].com.sv>

Estimado Sr. [redacted],

He tenido un mes bastante ajetreado pero quisiera retomar este tema, no como consultor, sino como cliente.

Quería expresarle que como cliente de [redacted], y responsable de la seguridad informática de una empresa de 400 empleados que utiliza su banco para cuentas corporativas y pagos de planilla me siento un poco defraudado de la poca importancia que se ha dado a este caso.



Desde hace varios meses el banco promociona intensivamente el sitio www.[redacted].com.sv

Inicialmente noté que el sitio no usaba un certificado SSL durante las transacciones ni a la hora de solicitar tarjeta de crédito como puede ver en el adjunto. Dejé un mensaje utilizando el campo de dirección de mi perfil indicando la falla y los contacté. Luego de unas 3 semanas recibí una llamada preguntando si había tenido problemas al usar el sitio, pero les pedí llamarme más tarde (lo cual no hicieron). El dia siguiente el sitio comenzó a usar un certificado. Esa es sin duda una mejora, pero hay aún problemas graves.

Como puede ver en el mismo adjunto, la URL tiene múltiples parámetros visibles a cualquier persona con acceso a las direcciones visitadas por el usuario (otros usuarios de la PC, administradores de proxys, logs empresariales). Este es un ejemplo de las URLs utilizadas por el sitio:

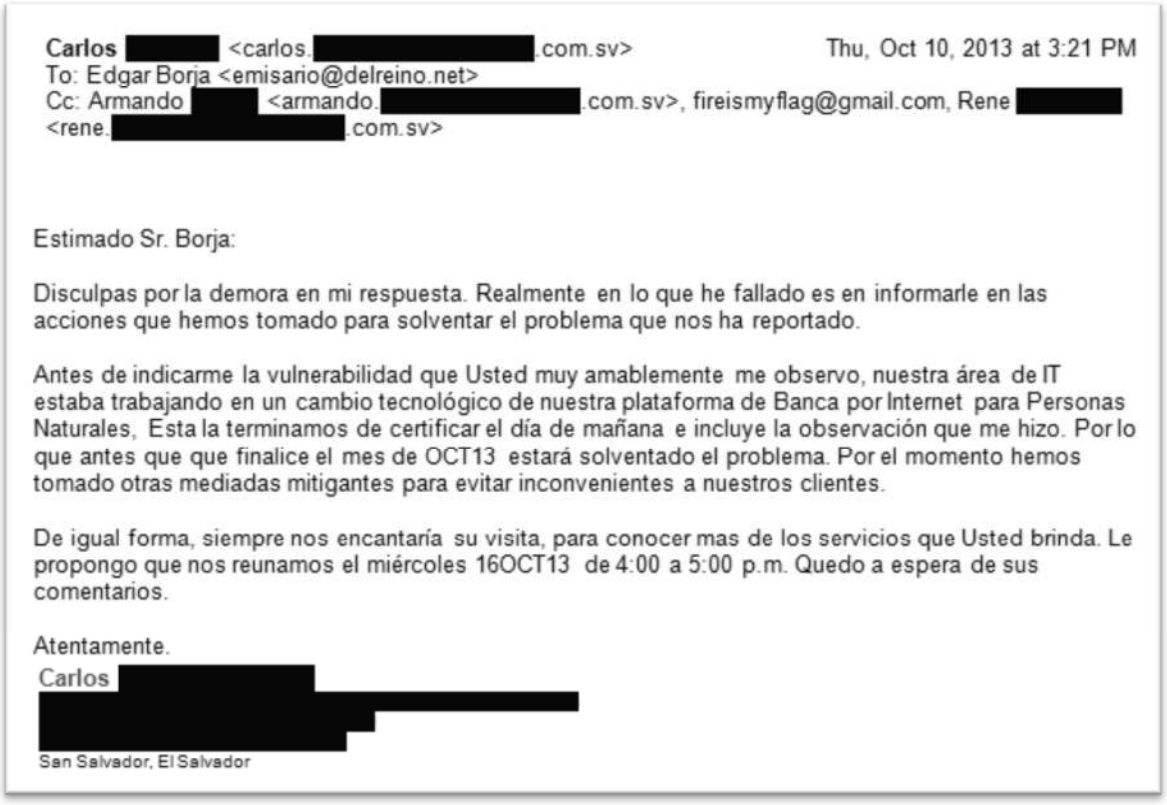
https://www.[redacted].com.sv/perfil.cfm?userid=15527&CFID=2619630&CFTOKEN=28778332

Según todas las pruebas que he realizado, esta URL llevará a cualquier persona directamente al perfil completo del cliente, incluyendo nombre completo, DUI, dirección, teléfono, fecha de nacimiento y **hasta la contraseña.** La única protección de la contraseña es que está en un campo tipo password, pero con sólo ver el código fuente puede leerse, o usando este código en la barra de dirección se puede lograr lo mismo:
javascript:alert(document.forms[1].registroPwd.value)
Esta es una indicación clara de otra falla grave del sitio: el almacenamiento de contraseñas no encriptadas; especialmente considerando que muchos de sus usuarios (incluyendo nuestros empleados) utilizarán la misma contraseña que tienen para su cuenta de banca electrónica.
Adicionalmente a la contraseña, todos los datos personales disponibles presentan un riesgo alto.
En mi caso, decidí no realizar ninguna transacción con mi tarjeta de crédito en este sitio, no sé si el número de trajeta de crédito queda guardado o sería visible para alguien que ganara acceso al perfil o contraseña del usuario.
Comprendo que este no es el sitio principal del banco, pero lo han promovido mucho y estoy seguro que muchos clientes están en riesgo con su información.
Les ruego que le den a este asunto la atención debida, y se tomen medidas para prevenir que nuestros empleados queden así de expuestos con nuevas implementaciones.
Atentamente,

Edgar Borja

[Quoted text hidden]

Luego de este nuevo mensaje recibimos otra respuesta del banco, donde manifestaban que sí estaban tratando el caso con diligencia y sólamente habían omitido notificarnos al respecto de sus avances.



Carlos [redacted] <carlos.[redacted].com.sv> Thu, Oct 10, 2013 at 3:21 PM
To: Edgar Borja <emisario@delreino.net>
Cc: Armando [redacted] <armando.[redacted].com.sv>, fireismyflag@gmail.com, Rene [redacted] <rene.[redacted].com.sv>

Estimado Sr. Borja:

Disculpas por la demora en mi respuesta. Realmente en lo que he fallado es en informarle en las acciones que hemos tomado para solventar el problema que nos ha reportado.

Antes de indicarme la vulnerabilidad que Usted muy amablemente me observo, nuestra área de IT estaba trabajando en un cambio tecnológico de nuestra plataforma de Banca por Internet para Personas Naturales, Esta la terminamos de certificar el día de mañana e incluye la observación que me hizo. Por lo que antes que que finalice el mes de OCT13 estará solventado el problema. Por el momento hemos tomado otras mediadas mitigantes para evitar inconvenientes a nuestros clientes.

De igual forma, siempre nos encantaría su visita, para conocer mas de los servicios que Usted brinda. Le propongo que nos reunamos el miércoles 16OCT13 de 4:00 a 5:00 p.m. Quedo a espera de sus comentarios.

Atentamente.
Carlos [redacted]
San Salvador, El Salvador

Luego de haber establecido la fecha de la reunión se procedió a preparar una demostración que expusiera claramente los problemas detectados, esta demostración se llevó a cabo en las oficinas del banco por parte de los dos miembros de nuestro equipo. Los encargados de sistemas del banco agradecieron el reporte que presentamos y se comprometieron a resolver los problemas detectados con prontitud, lo cual cumplieron oportunamente.

4.1.3 Análisis del resultado

Al revisar la secuencia de interacciones y el trabajo realizado por parte del banco, es evidente que tienen un proceso interno claro de reporte de vulnerabilidades, ya que aunque como personas externas contactamos a un área de atención al cliente, la información fue trasladada prontamente al área relevante, y hubo una persona encargada de llevar el seguimiento al incidente de principio a fin, esta persona estuvo apoyada por un equipo al que pudimos conocer durante la reunión final que se sostuvo en sus oficinas.

Otro de los hallazgos importantes fue que el sitio específico que sufría de las múltiples vulnerabilidades detectadas no fue desarrollado directamente por el banco, sino que fue desarrolado a través de una empresa consultora, sin embargo, la base de clientes general del banco se vio expuesta a los riesgos generados por los problemas detectados, y al evidenciar este tema en la reunión el banco se comprometió a establecer controles y pruebas de aceptación más estrictos con los consultores que trabajen en proyectos de desarrollo a los cuales puedan estar expuestos sus clientes.

Capítulo V: Conclusiones y recomendaciones

Existen una amplia variedad de posibles ataques y vulnerabilidades a los que una aplicación web puede estar expuesta, y nuevos ataques y métodos de protección surgen cada año; hace una década el panorama de los riesgos a los que se enfrentan los usuarios y los administradores de sitios web era diferente al actual, y no podemos predecir con certeza cuál será el estado de la seguridad de la información dentro de una década más.

Como pudimos observar a través de la investigación en todos los dominios .sv, no podemos garantizar que por alojar nuestro sitio con en un país del primer mundo estaremos protegidos por las prácticas óptimas de seguridad de la información, y la protección de las aplicaciones y los datos debe ser un proceso contínuo del ciclo de desarrollo y mantenimiento de aplicaciones, ya que aún en el caso de aplicaciones desarrolladas internamente, siempre existen componentes externos en la plataforma de trabajo, como el sistema operativo, el lenguaje de programación y el motor de base de datos; estos componentes son utilizados por múltiples proyectos, y están expuestos a que se les descubran vulnerabilidades, consecuentemente, toda solución de software, entre los que se pueden catalogar los sitios web, debe de mantenerse en un proceso de actualización con el fin de estar protegidos contra amenazas emergentes. (securedigitalsolutions, 2013)

Dos ejemplos claves en El Salvador fueron presentados en este documento: un sitio web público perteneciente a un reconocido banco, y el sitio de la organización que administra el dominio .sv a nivel nacional, ambos presentaban vulnerabilidades y ambos fueron notificados sobre los problemas detectados. En el caso del banco, existían riesgos graves para los usuarios ya que podrían haberse cometido fraudes financieros con la información no protegida del sitio.

Estos dos ejemplos poenen en evidencia que sin importar lo grande de una organización, la madurez de un programa de seguridad de la información siempre es un reto, y aún un sitio pequeño puede llegar a un nivel de seguridad que los que están disponibles para los sitios de una organización internacional, ya que la implementación técnica de un sitio seguro, y en un constante proceso de “hardening” están al alcance de cualquier empresa.

En el futuro, la seguridad de las aplicaciones web seguirá siendo un punto crítico para las empresas que funcionan en El Salvador y en el mundo, ya que cada vez más servicios se migran a esta plataforma, y se espera que en el futuro, algunos servicios estén disponibles únicamente para usuarios web; adicionalmente, nuevas tendencias como el uso de aplicaciones móviles refuerza la necesidad de una plataforma de software segura, ya que la mayoría de aplicaciones móbiles se apoyan en un servidor de servicios web para realizar sus transacciones.

Es importante que los profesionales y estudiantes de informática nos mantengamos al día de los nuevos descubrimientos en materia de seguridad de la información, protección de los datos, y tipos de ataques en la web, y principalmente las personas que son responsables de sitios web, con énfasis en los que tienen servicios financieros o comerciales, deben dedicar tiempo y esfuerzo a proteger la integridad de sus sistemas, ya que la reputación y el modus operandi de muchas organizaciones ahora dependen de su capacidad de hacer negocios en línea.

A través de un proceso de formación contínua para el personal de informática y de estandarización e implementación de procesos de seguridad de la información es posible reducir los riesgos y vulnerabilidades de los sitios y sistemas a niveles mínimos, y lograr un mayor grado de madurez y formalidad en las organizaciones.

Aun cuando los riesgos se minimicen es necesario reconocer que no pueden desaparecer completamente, ya que en la carrera por la seguridad de la información, muchas veces son los atacantes los que pueden tener la delantera al diseñar nuevos métodos de infiltración antes de que los fabricantes de software globales o las empresas de seguridad informática tengan tiempo de responder, por esto es necesario que toda empresa conozca con certeza dónde está su información, y también que implemente mecanismos de alerta y procesos de notificación que sean conocidos a través de toda la compañía, de forma que

los tiempos de respuesta a incidentes se reduzca, y en el caso de sufrir una pérdida de datos o un ataque sea posible medir con certeza el impacto potencial o real al negocio.

Uno de los activos más valiosos que puede ver dañada una empresa que sufra de un ataque informático es la reputación de la misma, si el incidente tiene un alto impacto y no se maneja apropiadamente puede afectar permanentemente a la organización.

Ya sea que un riesgo sea detectado durante un proceso de verificación interno del código de un sitio web, o que sea reportado por un cliente, o en el peor de los casos, que sea identificado luego de un ataque, es crítico que exista un proceso de aprendizaje que permita a los encargados mejorar contínuamente los procesos y sistemas existentes.

Considerando lo anterior, siempre será mejor que las vulnerabilidades se detecten internamente, y esto ocurrirá a través de procesos de prevención, como revisiones de código, con el uso de análisis automatizados, y el uso de escenarios de pruebas, por lo tanto presentamos como recomendación más importante el que estas prácticas que no siempre están presentes se vuelvan parte de la cotidianeidad del desarrollo de aplicaciones web en el contexto salvadoreño.

Referencias

Baker, Hillary. (2009). *Califonia State University.* Obtenido de http://www.csun.edu/it/information-security-incident-reporting

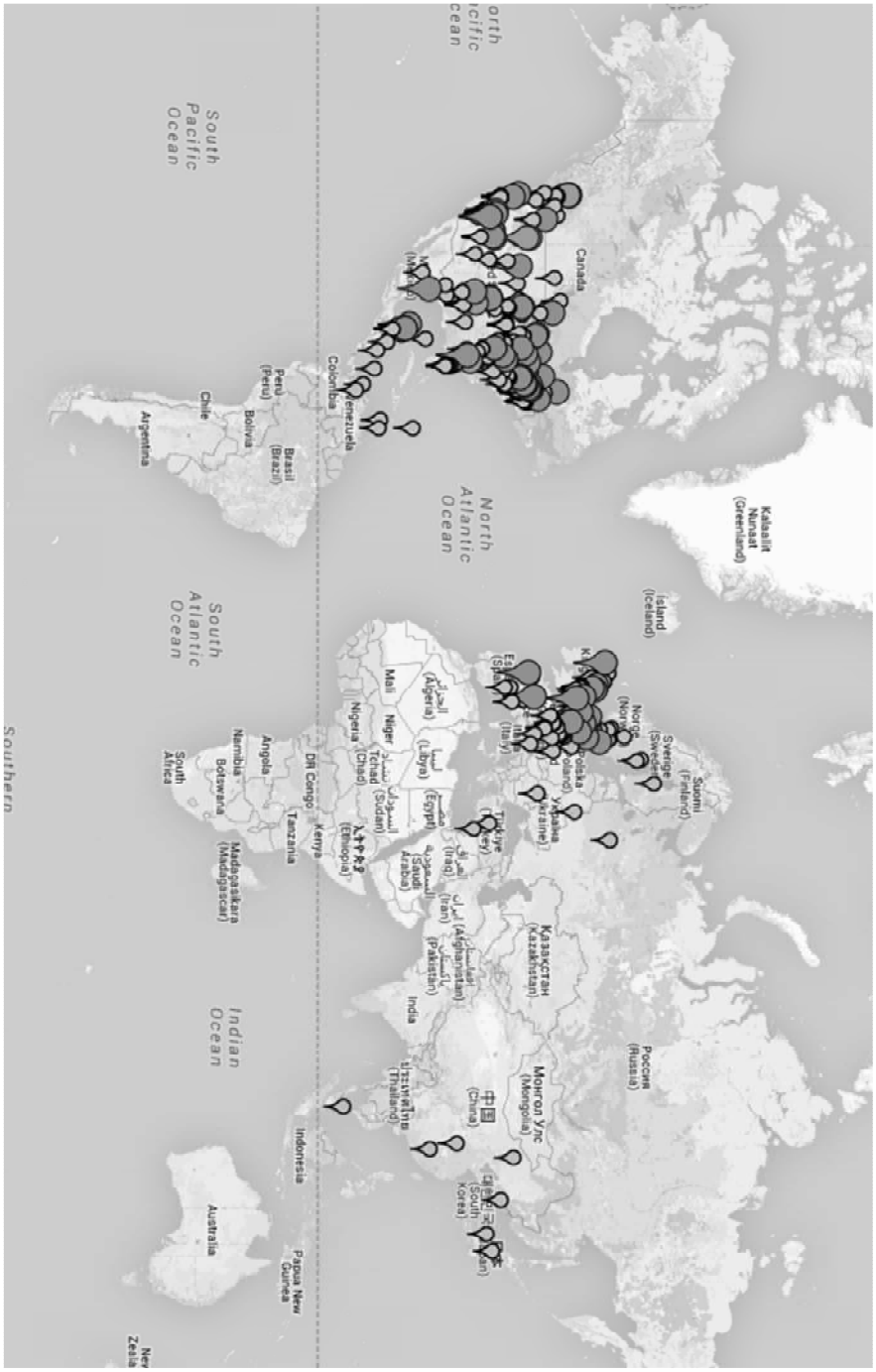
OWASP Foundation. (2013). *OWASP*. Obtenido de https://www.owasp.org

Riden, J. & McGeehan, R. *Know your Enemy.* Obtenido de http://www.honeynet.org/book/export/html/1

Secure Digital Solutions. (2013). *SECURE DIGITAL SOLUTIONS*. Obtenido de http://securedigitalsolutions.com/

Anexos

Mapa distribución de servidores para sitios .sv



Aplicación desarrollada para escanear dominios “.sv”

Clase de geolocalización

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.Linq;
using System.Globalization;
namespace ConsoleApplication1
{
    public class GeoLocation
    {
        public string CountryCode { get; set; }
        public string CountryName { get; set; }
        public string Region { get; set; }
        public string City { get; set; }
        public string ZipCode { get; set; }
        public decimal Latitude { get; set; }
        public decimal Longitude { get; set; }

        public static GeoLocation GeoLocate(string ip)
        {
            const string API_KEY = "5733bfaf1ab05b0a5baf6bd030dc31ba9a0f06b987432ccdd1743772172edcdf";

            var url = "http://api.ipinfodb.com/v3/ip-city/?key={0}&ip={1}&format=xml";

            var doc = XDocument.Load(string.Format(url, API_KEY, ip));

            return new GeoLocation
            {
                CountryCode = doc.Descendants("countryCode").First().Value,
                CountryName = doc.Descendants("countryName").First().Value,
                Region = doc.Descendants("regionName").First().Value,
                City = doc.Descendants("cityName").First().Value,
                ZipCode = doc.Descendants("zipCode").First().Value,
                Latitude = decimal.Parse(doc.Descendants("latitude").First().Value,
CultureInfo.InvariantCulture),
                Longitude = decimal.Parse(doc.Descendants("longitude").First().Value,
CultureInfo.InvariantCulture)
            };
        }
    }
}
```

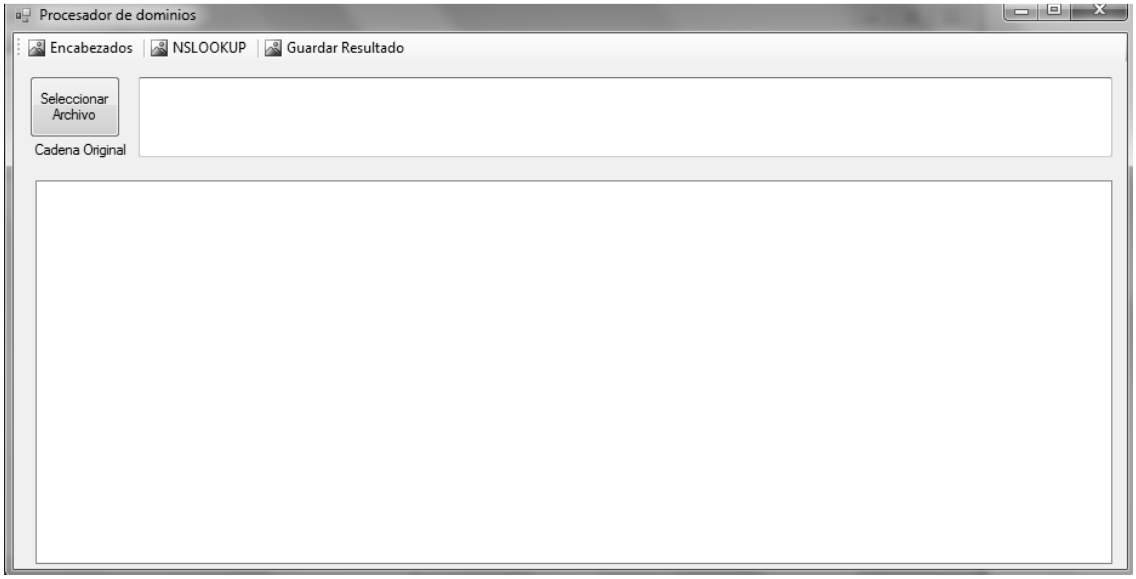
Clase de detección de dirección IP e implementación de geolocalización.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.IO;
using System.Data.SqlClient;
namespace ConsoleApplication1
{
    public partial class Form1 : Form
    {
        delegate void AddLsItem(string str);
        public Form1()
        {
            InitializeComponent();
        }
        Domains dms;
        DateTime FechaProc;
        private void button1_Click(object sender, EventArgs e)
        {
            DomainManager dm = new DomainManager(this.txtOriginalString.Text);
            dms = dm.Dominios();
            this.FechaProc = DateTime.Now;
            Thread rn = new Thread(Scaneo);
            rn.Start();
        }
        public void Scaneo() {
            foreach (Pack obj in dms.Domais) {
                //Thread hilo = new Thread(() => ScanUnico(obj));
                //hilo.Start();
                ScanUnico(obj);
                StreamWriter srt = new StreamWriter("c:\\scan\\ips.txt");
                foreach (var et in this.lsResultado.Items) {
                    srt.WriteLine(et.ToString());
                }
                srt.Close();
                srt.Dispose();
            }
        }
        public void ScanUnico(Pack obj) {
            try
            {
                try
                {
                    NetHelper help = new NetHelper(obj.col1.Trim());
                    string Item = string.Format("Estado:{0}|Dominio:{1}|IPs:{2}|Alias:{3}|MSG:{4}|Extra:{5}",
"OK", obj.col1, help.GetArrIP(), help.GetIpAlias(),
                        "", obj.col2);
                    GeoLocation geo = GeoLocation.GeoLocate(help.GetArrIP());

                    RegistrarResultado(obj.col1, obj.col2, help.GetIpAlias(), help.GetArrIP(), help.GetArrIP(),
false,
                        "", geo.CountryCode, geo.CountryName, geo.Region, geo.City, geo.ZipCode, geo.Latitude,
geo.Longitude, FechaProc);

                    this.AddItem(Item);
                }
                catch (Exception es) {
                    NetHelper help = new NetHelper("www." + obj.col1.Trim());
                    string Item = string.Format("Estado:{0}|Dominio:{1}|IPs:{2}|Alias:{3}|MSG:{4}|Extra:{5}",
"OK", "www." + obj.col1, help.GetArrIP(), help.GetIpAlias(),
                        "", obj.col2);
```

```
                    GeoLocation geo = GeoLocation.GeoLocate(help.GetArrIP());

                    RegistrarResultado("www." + obj.col1, obj.col2, help.GetIpAlias(), help.GetArrIP(),
help.GetArrIP(), false,
                        "", geo.CountryCode, geo.CountryName, geo.Region, geo.City, geo.ZipCode, geo.Latitude,
geo.Longitude, FechaProc);

                    this.AddItem(Item);

                }
            }
            catch (Exception ex)
            {
                this.AddItem(string.Format("Estado:{0}|Dominio:{1}|IPs:|Alias:|MSG:{2}|Extra:{3}","ERROR",
obj.col1, ex.ToString(),obj.col2));

                RegistrarResultado(obj.col1, obj.col2, obj.col1, "", "", true, ex.Message, "", "", "", "", "",
0, 0, FechaProc);
            }
        }

        public void RegistrarResultado(string dominio, string domname, string alias, string ip1, string ip2,
bool error,
            string msg, string contrycode, string contryname, string region, string city, string zipcode,
decimal latitud, decimal logitud
            , DateTime fechaproc)
        {
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=DireccionesIp;Integrated
Security=True");
            con.Open();
            string sql1 = "delete from [Domains] where Dominio=@Dominio; ";
            string sql =sql1+ "insert into [Domains](Dominio ,DominioName ,Alias ,"
            + " DireccionIP1 ,DireccionIP2 ,Error ,Msg ,CountryCode ,CountryName ,"
            + " Region ,City ,ZipCode ,Latitud ,Longitud ,FechaProc)"
            + " values (@Dominio ,@DominioName ,@Alias ,"
            + " @DireccionIP1 ,@DireccionIP2 ,@Error ,@Msg ,@CountryCode ,@CountryName ,"
            + " @Region ,@City ,@ZipCode ,@Latitud ,@Longitud ,@FechaProc);";
            SqlCommand comna = new SqlCommand(sql, con);
            comna.Parameters.AddWithValue("@Dominio", dominio);
            comna.Parameters.AddWithValue("@DominioName", domname);
            comna.Parameters.AddWithValue("@Alias", alias);
            comna.Parameters.AddWithValue("@DireccionIP1", ip1);
            comna.Parameters.AddWithValue("@DireccionIP2", ip2);
            comna.Parameters.AddWithValue("@Error", error);
            comna.Parameters.AddWithValue("@Msg", msg );
            comna.Parameters.AddWithValue("@CountryCode", contrycode);
            comna.Parameters.AddWithValue("@CountryName", contryname);
            comna.Parameters.AddWithValue("@Region", region);
            comna.Parameters.AddWithValue("@City", city);
            comna.Parameters.AddWithValue("@ZipCode", zipcode);
            comna.Parameters.AddWithValue("@Latitud", latitud);
            comna.Parameters.AddWithValue("@Longitud", logitud);
            comna.Parameters.AddWithValue("@FechaProc", fechaproc);
            comna.ExecuteNonQuery();
            con.Close();

        }
        public void AddItem(string item) {
            if (this.lsResultado.InvokeRequired)
            {
                AddLsItem d = new AddLsItem(AddItem);
                this.Invoke(d, new object[] { item });
            }
            else {
                this.lsResultado.Items.Add(item);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            FrmLecturaEstado frm = new FrmLecturaEstado();
            frm.Show();
        }
```

```
        private void button3_Click(object sender, EventArgs e)
        {
            SaveFileDialog sv = new SaveFileDialog();
            if (sv.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                StreamWriter sw = File.CreateText(sv.FileName);
                foreach(var ob in this.lsResultado.Items){
                    sw.WriteLine(ob.ToString());
                }
                sw.Close();
                //File.WriteAllText(sv.FileName, this.txtComentario.Text);

            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            OpenFileDialog op = new OpenFileDialog();
            if (op.ShowDialog() == System.Windows.Forms.DialogResult.OK) {
                this.txtOriginalString.Text = File.OpenText(op.FileName).ReadToEnd();
            }
        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            FrmWebRequest frm = new FrmWebRequest();
            frm.Show();
        }

        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            DomainManager dm = new DomainManager(this.txtOriginalString.Text);
            dms = dm.Dominios();
            this.FechaProc = DateTime.Now;
            Thread rn = new Thread(Scaneo);
            rn.Start();
        }

        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            SaveFileDialog sv = new SaveFileDialog();
            if (sv.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                StreamWriter sw = File.CreateText(sv.FileName);
                foreach (var ob in this.lsResultado.Items)
                {
                    sw.WriteLine(ob.ToString());
                }
                sw.Close();
                //File.WriteAllText(sv.FileName, this.txtComentario.Text);

            }
        }
    }
}
```

Interfaz de usuario, geolocalización y detección de direcciones IP



Clase de escaneos de encabezados.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.IO;
using System.Data;
using System.Data.SqlClient;
namespace ConsoleApplication1
{
    public partial class FrmWebRequest : Form
    {
        public FrmWebRequest()
        {
            InitializeComponent();
        }
        delegate void Progreso(int estado, int index);
        private void Procesar(DataGridViewRow direccion) {

            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=DireccionesIp;Integrated
Security=True");
            con.Open();
            try {
                SqlCommand sqlClean = new SqlCommand("delete  from dbo.Directiro where dominio=@Dominio", con);
                sqlClean.Parameters.AddWithValue("@Dominio", direccion.Cells["Dominio"].Value);
                sqlClean.ExecuteNonQuery();
```

```
                HttpWebRequest myHttpWebRequest = (HttpWebRequest)WebRequest.Create(string.Format("http://{0}",
direccion.Cells["DireccionIP1"].Value));
                // Sends the HttpWebRequest and waits for response.
                HttpWebResponse myHttpWebResponse = (HttpWebResponse)myHttpWebRequest.GetResponse();

                // Displays all the headers present in the response received from the URI.
                Console.WriteLine("\r\nThe following headers were received in the response:");
                // Displays each header and it's key associated with the response.


                for (int i = 0; i < myHttpWebResponse.Headers.Count; ++i)
                {

                    string sql = "insert into
Directiro(Dominio,DireccionIP1,DireccionIP2,Header,Valor,Error,Msg )values("
                         + "@Dominio,@DireccionIP1,@DireccionIP2,@Header,@Valor,@Error,@Msg)";
                    SqlCommand comna = new SqlCommand(sql, con);
                    comna.Parameters.AddWithValue("@Dominio", direccion.Cells["Dominio"].Value);
                    comna.Parameters.AddWithValue("@DireccionIP1", direccion.Cells["DireccionIP1"].Value);
                    comna.Parameters.AddWithValue("@DireccionIP2", direccion.Cells["DireccionIP1"].Value);
                    comna.Parameters.AddWithValue("@Header", myHttpWebResponse.Headers.Keys[i]);
                    comna.Parameters.AddWithValue("@Valor", myHttpWebResponse.Headers[i]);
                    comna.Parameters.AddWithValue("@Error", false);
                    comna.Parameters.AddWithValue("@Msg", "");
                    comna.ExecuteNonQuery();
                    //MessageBox.Show(String.Format("\nHeader Name:{0}, Value :{1}",
myHttpWebResponse.Headers.Keys[i], myHttpWebResponse.Headers[i]));
                }

                // Releases the resources of the response.
                myHttpWebResponse.Close();

            }
            catch (Exception ex) {
                string sql = "insert into  Directiro(Dominio,DireccionIP1,DireccionIP2,Header,Valor,Error,Msg
)values("
                            + "@Dominio,@DireccionIP1,@DireccionIP2,@Header,@Valor,@Error,@Msg)";
                SqlCommand comna = new SqlCommand(sql, con);
                comna.Parameters.AddWithValue("@Dominio", direccion.Cells["Dominio"].Value);
                comna.Parameters.AddWithValue("@DireccionIP1", direccion.Cells["DireccionIP1"].Value);
                comna.Parameters.AddWithValue("@DireccionIP2", direccion.Cells["DireccionIP1"].Value);

                comna.Parameters.AddWithValue("@Header", "");
                comna.Parameters.AddWithValue("@Valor", "");
                comna.Parameters.AddWithValue("@Error", true);
                comna.Parameters.AddWithValue("@Msg", ex.Message);
                comna.ExecuteNonQuery();
            }

        }
        private void button2_Click(object sender, EventArgs e)
        {
            OpenFileDialog fls = new OpenFileDialog();
            if (fls.ShowDialog() == System.Windows.Forms.DialogResult.OK) {
                StreamReader fl = File.OpenText(fls.FileName);
               // this.textBox2.Text = fl.ReadToEnd();
                fl.Close();
            }
        }
        DataSet ds = new System.Data.DataSet();
        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            this.txtFiltro.Text = "";
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=DireccionesIp;Integrated
Security=True");
            SqlDataAdapter da = new SqlDataAdapter("select * from dbo.Domains", con);
            con.Open();
            ds.Clear();
            da.Fill(ds);
            this.dataGridView1.DataSource = ds.Tables[0];
            CargarResumen();
        }

        private void toolStripButton2_Click(object sender, EventArgs e)
```

```
        {
            this.brProgres.Maximum = this.dataGridView1.Rows.Count;
            this.brProgres.Value = 0;
            System.Threading.Thread tr = new System.Threading.Thread(ProcesarLote);
            tr.Start();
        }

        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            try
            {
                SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=DireccionesIp;Integrated
Security=True");
                SqlDataAdapter da = new SqlDataAdapter(string.Format("select * from dbo.Domains {0}",
this.txtFiltro.Text != "" ? " where " + this.txtFiltro.Text : ""), con);
                con.Open();
                ds.Clear();
                da.Fill(ds);
                this.dataGridView1.DataSource = ds.Tables[0];
                CargarResumen();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message);
            }
        }
        private void ProcesarLote() {
            DataGridViewRowCollection drs = this.dataGridView1.Rows;
            int actual = 1;
            foreach (DataGridViewRow dr in drs)
            {
                if ((Boolean)dr.Cells["Error"].Value == false)
                {
                    Procesar(dr);
                    setProgres(actual,dr.Index);
                    actual++;
                }
            }

        }
        private void setProgres(int actual, int index) {
            if (brProgres.ProgressBar.InvokeRequired)
            {
                Progreso pr = new Progreso(setProgres);
                this.Invoke(pr, new object[] { actual, index });
            }
            else {
                brProgres.Value = actual;
                dataGridView1.Rows[index].DefaultCellStyle.BackColor = Color.Gray;
            }

        }

        private void CargarResumen() {
            string sql = "select Header, Valor, count(*) from dbo.Directiro Cantidad  where valor !='' "
            + " and header in ('Cache-Control','Server','X-Powered-By')"
            + " group by Header, Valor"
            + " order by header";
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=DireccionesIp;Integrated
Security=True");
            SqlDataAdapter da = new SqlDataAdapter(sql, con);
            con.Open();

            DataSet ds1 = new System.Data.DataSet();
            da.Fill(ds1);
            this.dataGridView3.DataSource = ds1.Tables[0];

        }

        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            DataGridViewRow dr = this.dataGridView1.Rows[e.RowIndex];
            string sql = "select * from dbo.Directiro where dominio='" + dr.Cells["Dominio"].Value.ToString() +
"'";
```

```
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=DireccionesIp;Integrated
Security=True");
            SqlDataAdapter da = new SqlDataAdapter(sql, con);
            con.Open();

            DataSet ds1 = new System.Data.DataSet();
            da.Fill(ds1);
            this.dataGridView2.DataSource = ds1.Tables[0];
        }

    }
}
```