ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

DESARROLLO DEL PRODUCTO PARA TEST DE PENETRACION ENFOCADO EN EL FUZZING DE APLICACIONES

TESIS DE GRADO

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES
ESPECIALIZACION SISTEMAS DE INFORMACION

INGENIERO EN CIENCIAS COMPUTACIONALES
ESPECIALIZACION SISTEMAS MULTIMEDIA

INGENIERO EN CIENCIAS COMPUTACIONALES
ESPECIALIZACION SISTEMAS DE INFORMACION

Presentado por:

López Sánchez Omar Felipe

Ochoa Samaniego Ingrid Alfonsina

Pibaque Suárez Angélica María

Guayaquil - Ecuador

2011

AGRADECIMIENTO

Nuestro más Sincero Agradecimiento a Dios por guiarnos en la toma de nuestras decisiones, por brindarnos cada día nuevas oportunidades, a nuestros Padres por ser el ejemplo a seguir, por apoyarnos siempre en los estudios, a nuestros hermanos por ayudarnos en todo lo que podían.

TRIBUNAL SUSTENTACIÓN

Ing. Alfonso Aranda
PROFESOR DE LA MATERIA DE GRADUACIÓN

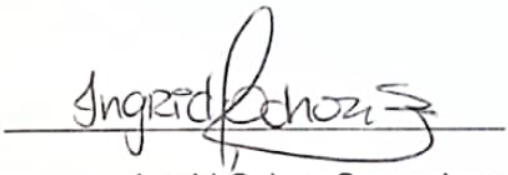
Ing. Vanessa Cedeño
PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

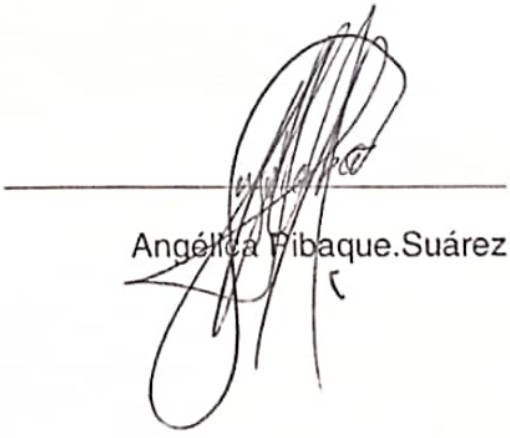
"La responsabilidad del contenido de este informe de materia de graduación, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Omar López Sánchez.



Ingrid Ochoa Samaniego



Angélica Pibaque.Suárez

RESUMEN

Vivimos en plena era tecnológica, donde cada vez más información y datos de carácter personal son informatizados. La seguridad de los sistemas toma mayor importancia, sobre todo ahora que hay más amenazas y vulnerabilidades es necesario proteger uno de los activos más importantes de la organización, la información, garantizando siempre la disponibilidad, la confidencialidad e integridad de la misma.

La forma más adecuada para proteger dichos activos es mediante una correcta gestión del riesgo, logrando así identificar y focalizar esfuerzos hacia aquellos elementos que se encuentren más expuestos con el fin de detectar defectos en los sistemas.

Nuestro fin es dar a conocer que mediante la implementación y mejora de herramientas que sean capaces de categorizar la veracidad de las alertas y descartar los fallos de las aplicaciones, se podrá garantizar a la organización a que adopte las buenas prácticas sugeridas por la ISO27001:2005 para un correcto tratamiento del riesgo.

ÍNDICE GENERAL

3

4

5

ÍNDICE DE FIGURAS

GLOSARIO DE TÉRMINOS

BUGS: Fallas técnicas que se detectan cuando el elemento tecnológico, como hardware o software, ya se encuentra en producción.

BUFFER OVERFLOW: Es un error de software que se produce cuando se copia una cantidad de datos sobre un área que no es lo suficientemente grande para contenerlos, sobrescribiendo de esta manera otras zonas de memoria. Esto se debe en general a un fallo de programación.

OWASP (Open Web Application Security Project): Proyecto Abierto de Seguridad de Aplicaciones Web.

XSS (Cross Site Scripting): Son ataques de inyección de código en los diferentes
intérpretes del navegador web. Se pueden llevar a cabo utilizando HTML, Java Script, VBScript, ActiveX, Flash y otros lenguajes del lado del cliente.

UNIX: es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

VERITAS BACKUP EXEC: Software desarrollado por Symantec

SETUID: términos de Unix, abreviatura para "Set User ID".

SDLC (SOFTWARE DEVELOPMENT LIFE CYCLE): Ciclo de Vida de Desarrollo del Software.

FRAMEWORK: termino escogido para la traducción: *entorno de trabajo*.

SERVICIOS PÚBLICOS: Las actividades, entidades u órganos públicos o privados con personalidad jurídica creados por Constitución o por ley, para dar satisfacción en forma regular y continua a cierta categoría de necesidades de interés general, bien en forma directa, mediante concesionario o a través de cualquier otro medio legal con sujeción a un régimen de Derecho Público o Privado, según corresponda".

PRESCRIPTORES: También llamados indicadores son aquellos que conociendo el producto pueden influir por diferentes motivos en la adquisición o no de un bien determinado.

VAN: Es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión. La metodología consiste en descontar al momento actual (es decir, actualizar mediante una tasa) todos los flujos de caja futuros del proyecto. A este valor se le resta la inversión inicial, de tal modo que el valor obtenido es el valor actual neto del proyecto.

TIR: La tasa interna de retorno o tasa interna de rentabilidad de una inversión, está definida como la tasa de interés con la cual el valor actual neto o valor presente neto (VAN) es igual a cero. Es un indicador de la rentabilidad de un proyecto, a mayor TIR, mayor rentabilidad.

INTRODUCCIÓN

Actualmente, las tecnologías han revolucionado al mundo, mostrando nuevas y diferentes formas de manejar la información, se considera a la información como un activo de la empresa, por lo tanto es primordial proteger este activo, considerado uno de los más importantes de las empresas.

Para proteger este activo las empresas buscan nuevas soluciones que no solo vayan de la mano con el desarrollo de la tecnología sino que también los provea de forma eficiente y eficaz que su información esté disponible en todo momento.

Existen en este momento diferentes implementaciones que realizan esta función pero que por barreras tecnológicas aun se desconocen en nuestro medio ya sea por desinformación o por cuestiones económicas todavía no se han desarrollado.

Las técnicas de pruebas de vulnerabilidades son utilizadas en la actualidad en el mercado local pero aun no son completamente explotadas, es por esta razón que se decidió desarrollar el producto Test de Penetración, que no solo permite realizar pruebas de vulnerabilidades, sino que a través del Fuzzing de Aplicaciones podemos hacer un seguimiento a profundidad sobre las vulnerabilidades detectadas.

Esta técnica puede estar enfocada en varios instancias desde protocolos de red, formatos de archivos, sistemas de ficheros, etc. Siendo de gran utilidad no solo para los auditores informáticos sino también para los desarrolladores de software que tendrían a esta herramienta como un apoyo.

Es aquí donde nuestro producto encuentra un mercado objetivo que con una campaña de información puede llegar a ser una solución para las empresas, ya que de esta manera protegerían su activo más importante.

1. ANTECEDENTES Y JUSTIFICACION

1.1. Antecedentes

La evolución tecnológica genera oportunidades pero también grandes riesgos a los activos de información, uno de los bienes más valiosos. Hay que garantizar que los recursos informáticos de una compañía estén disponibles para cumplir sus propósitos, es decir, que no estén dañados o alterados por circunstancias o factores externos, es una definición útil para conocer lo que implica el concepto de seguridad informática.

En la actualidad la seguridad informática es usada para proteger la confidencialidad, disponibilidad e integridad de la información. Es importante señalar que su manejo está basado en la tecnología y que a medida que ésta va avanzando afecta su seguridad.

Además, la seguridad de la información involucra la implementación de estrategias que cubran los procesos en donde la información es el activo primordial. Estas estrategias deben tener como punto primordial el establecimiento de políticas, controles de seguridad, tecnologías y procedimientos para detectar amenazas que puedan explotar vulnerabilidades y que pongan en riesgo dicho activo, es decir, que ayuden a proteger y salvaguardar tanto información como los sistemas que la almacenan y administran

1.2. Objetivos

Actualmente no se utilizan métodos de evaluación de la seguridad de un sistema informático, la información debe ser protegida desde antes de lanzar un nuevo producto al mercado encontrando los fallos de seguridad, por medio de varias herramientas nosotros nos enfocaremos en un producto que analice y encuentre estos fallos por medio de un test de penetración enfocado en el fuzzing de aplicaciones.

1. El proyecto consiste en la oferta, al mercado nacional e internacional, de un servicio de análisis de seguridades de red y aplicaciones basado en técnicas de fuzzing las mismas que consisten no sólo identificar huecos de seguridad a nivel de aplicación sino también identificar la tolerancia a fallas, bugs desconocidos, capacidad de procesamiento, corrupción de memoria y errores en general. El desarrollo de este proyecto se enfocará en los siguientes puntos:
2. Desarrollar y ejecutar la estrategia de ventas, publicidad y marketing del Servicio (Brochure, Sitios Web, Lista de precios, descripción técnica)
3. Empaquetar y customizar el Kit de herramientas que se usarán para lo cual se hará un estudio del arte de los recursos existentes, para plantear y validar mejoras.
4. Desarrollar la metodología que categorice la veracidad de las alertas para así descartar falsos positivos de todos los reportes que arrojen las herramientas.
5. Profesionalización del reporte, en diferentes niveles: Técnico detallado, Técnico y Gerencial.

1.3. Introducción

1.3.1. Justificación

Con el avance de la tecnología y a su vez de las amenazas, la seguridad de los sistemas toma progresivamente mayor importancia, sobre todo ahora que la mayoría de los atacantes ya no se contentan con hacer acto de presencia, sino que son contratados por empresas de la competencia para saltarse las medidas de protección y robar o destruir datos importantes.

Existen diversas formas de intrusión en un sistema, desde el acceso físico al remoto, aprovechándose este último de los servicios ofrecidos por las máquinas. Normalmente se piensa que con un buen antivirus, un cortafuegos con buenas reglas de filtrado y una buena política de parcheo es suficiente. Pero nadie se para a pensar en las llamadas vulnerabilidades, de las que nadie tiene conocimiento y que se descubren a diario en cantidad de aplicaciones.

Para lograr el descubrimiento de estos fallos se suelen emplear, entre otras, unas herramientas que se dedican a lanzar peticiones mal formadas de forma automática. A éstas se les llama fuzzers, y pueden estar enfocadas tanto a protocolos de red, como a formatos de archivos, sistemas de ficheros, etc. Son de gran utilidad para auditores informáticos, estos evitan intrusiones indeseadas y desarrolladores de software obtienen un producto seguro.

1.4. Estado de Arte

En la actualidad la seguridad de la información es usada para proteger la confidencialidad de la información, ya que ésta tiene un alto valor y puede ser divulgada, mal utilizada o borrada. Es importante señalar que su manejo está basado en la tecnología y que a medida que ésta va avanzando afecta su disponibilidad y la pone en riesgo.

Además, la seguridad de la información involucra la implementación de estrategias que cubran los procesos en donde la información es el activo primordial. Estas estrategias deben tener como punto primordial el establecimiento de políticas, controles de seguridad, tecnologías y procedimientos para detectar amenazas que puedan explotar vulnerabilidades y que pongan en riesgo dicho activo, es decir, que ayuden a proteger y salvaguardar tanto información como los sistemas que la almacenan y administran.

1.5. Alcance y Restricciones

En este proyecto se implementará un fuzzer basado en diferentes técnicas de testeo de aplicaciones web que sean capaces de generar datos de distintos tamaños y enviar estos a uno de los puntos de entrada existentes en la aplicación, con el objeto de observar su comportamiento ante estas pruebas o mejor dicho como estos requerimientos son manejados. Si el programa a testear falla en los resultados del análisis, esto significará dos cosas: que hay defectos que corregir, y que probablemente sea posible explotar la falla.

El proceso empezará en identificar las entradas, tales como páginas web, directorios, campos ocultos, encabezados HTTP y cookies, el medio más sencillo y eficaz de identificar algún fallo implica simplemente en navegar por la página web y revisar el código fuente de la misma.

Lo más importante a remarcar es que estas técnicas son genéricas – es decir, que no están diseñadas para un código específico, sino para aplicaciones en general. Lo que significa que aunque se pueden encontrar algunos problemas genéricos, no tienen el conocimiento suficiente sobre una aplicación como para permitirles detectar la mayoría de los fallos.

2. MARCO TEORICO

2.1. ¿Qué es Fuzzing?

Se conoce como fuzzing a las diferentes técnicas de pruebas de software capaces de generar y enviar datos secuenciales o aleatorios a una o varias áreas o puntos de una aplicación, con el objeto de detectar defectos o vulnerabilidades existentes en el software auditado.

Generalmente utilizado como complemento a las prácticas habituales de chequeo de software, ya que proporcionan cobertura a fallos de datos y regiones de código no testados, gracias a la combinación del poder de la aleatoriedad y ataques heurísticos entre otros.

El fuzzing es usado por compañías de software y proyectos Open Source para mejorar la calidad del software, por investigadores de seguridad para descubrir y publicar vulnerabilidades, por auditores informáticos para analizar sistemas, y, en última instancia, por delincuentes informáticos (hackers) para encontrar agujeros en sistemas y explotarlos de forma secreta.

2.2. ¿Qué es Test de Penetración?

El Test de Penetración, también llamado a veces “hacking ético” es una evaluación activa de las medidas de seguridad de la información. En los entornos de red complejos actuales, la exposición potencial al riesgo es cada vez mayor y hacer a los sistemas seguros se convierte en un auténtico

reto.

A través del Test de Penetración es posible detectar el nivel de Seguridad Interna y Externa de los Sistemas de Información de la empresa, determinando el grado de acceso que tendría un atacante con intenciones maliciosas. Además, el servicio chequea las vulnerabilidades que pueden ser vistas y explotadas por individuos no autorizados, "hackers", agentes de información, ladrones, antiguos empleados, competidores, etc. Estos servicios permiten:

- Evaluar vulnerabilidades por medio de la identificación de debilidades de configuración que puedan ser explotadas.
- Analizar y categorizar las debilidades explotables basadas en el impacto potencial y posibilidad de ocurrencia.
- Proveer recomendaciones para mitigar y eliminar las debilidades.

El Test de Penetración está dirigido a la búsqueda de agujeros de seguridad de forma focalizada en uno o varios recursos críticos, como puede ser el firewall o el servidor Web.

2.3. Historia

La referencia más temprana al termino fuzzing aparece alrededor de 1989, el Profesor Barton Miller (considerado el padre del fuzzing) y su clase Advanced Operating System, se desarrollo y uso un fuzzer primitivo para testear aplicaciones UNIX. El objetivo de este test no era necesariamente el acceso a la seguridad del sistema, sino a la calidad y fiabilidad del código.

En 1995 la prueba fue repetida pero expandida hacia las utilidades de UNIX y pruebas del sistema operativo.

El método de fuzzing empleado por el equipo de Miller era muy brusco, si la aplicación colapsaba la prueba fallaba, sino terminaba con éxito. Esto se logro con simples cadenas aleatorias de caracteres objetivos de la aplicación.

Alrededor de 1999, comenzó a trabajar en la Universidad de Oulu en PROTOS, varias de estas pruebas fueron desarrolladas por protocolos de análisis y paquetes que no cumplían con las respectivas especificaciones.

En el 2002 Microsoft fundo la iniciativa PROTOS y en el 2003 los miembros del equipo de PROTOS lanzaron Codenomicon, una compañía dedicada al diseño y producción de pruebas comerciales de fuzzing. Aunque el producto está basado en la idea original de Oulu pero incluye una interfaz gráfica, soporte de usuario, entre otros.

En 2002 Dave Aitel lanzo en Open Source la herramienta llamada SPIKE bajo licencia GNU, este implementa un bloque que se acerca a las aplicaciones para testing – network, toma algunos elementos de las pruebas de Miller, pero lo más notable es que incluye la habilidad para describir el tamaño de las variables de los bloques de datos, no pueden generar datos aleatorios pero también empaqueta librerías de valores como las que producen fallos en aplicaciones pobremente escritas.

La mayoría de innovaciones que fueron lanzadas eran en forma de herramientas para las diferentes clases de fuzzing. En el 2004 Common Gateway Interface (CGI) script fue diseñado para producir archivos HTML malformados que se repetían cuando se refrescaba un navegador.

En el 2005 la compañía Mu Security comenzó a desarrollar un Hardware fuzzing para mutar protocolos de datos en una red. Active X comenzó a ser popular en el 2006 cuando David Zimmer lanzo COMRaider y HD Moore publico AxMan, ambas herramientas trabajaban con controles ActiveX que podían ser instanciados por aplicaciones web y con el navegador de

Microsoft, Internet Explorer. Remotamente explotaban vulnerabilidades que representaban un riesgo en el uso.

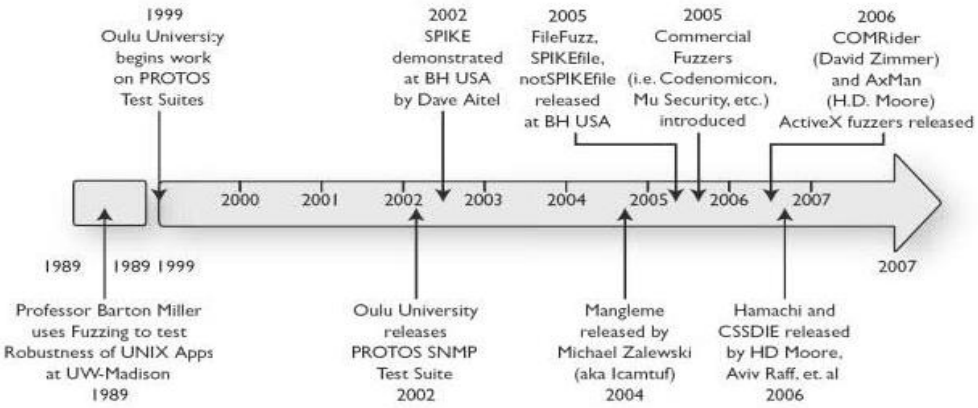


Figura 2.1 Historia del Fuzzing

2.4. Fases del Fuzzing

2.4.1. Descripción

Esto depende directamente de la aplicación objetivo, los parámetros de búsqueda y el formato de datos a ser usado, Sin embargo hay una metodología básica que podemos aplicar:

Identificación del objetivo

No es posible seleccionar una herramienta o técnica hasta tener el objetivo en mente. Si se va a desarrollar una aplicación durante una auditoria informática, el objetivo debe ser seleccionado con cuidado. Sin embargo si se buscan vulnerabilidades en una aplicación se puede tener más flexibilidad. Más allá de seleccionar aplicación, podría ser necesario identificar archivos específicos o librería dentro de la aplicación si este es

el caso se podría consultar archivos binarios de múltiples aplicaciones, así como vulnerabilidades de alto riesgo expandidas por usuarios bases.

Identificación de entradas

Virtualmente todas las vulnerabilidades son explotadas debido a que las aplicaciones aceptan las entradas y procesos de datos sin las correctas validaciones rutinarias.

Enumerar vectores de entrada es un proceso básico para el fuzzing. Al fallar códigos potenciales de entradas o entradas inesperadas pueden ser una limitante para las pruebas.

Se deben enviar desde los clientes datos como cabeceras, nombres de archivos, variables, registros, etc. Todo debe ser considerado como potencial variable para el fuzzing.

Generar datos fuzzed

Una vez identificados las entradas, los datos fuzzer deben ser generados. La decisión de usar valores predeterminados, datos mutados o datos generados dinámicamente depende del objetivo y el formato de datos, generalmente este proceso debe ser automatizado.

Ejecutar datos fuzzed

Este paso va de la mano con el paso previo, la ejecución envuelve el acto de enviar paquetes de datos hacia el objetivo, abrir archivos, y procesos, de nuevo esto es automatizado, sin esto no se puede realizar el fuzzing.

Monitor para excepciones

Eso es vital pero frecuentemente pasado por alto durante el fuzzing, el monitoreo del proceso, por ejemplo si se transmiten 10000 paquetes fuzzer a un servidor web esto puede hacer colapsar al servidor, al

monitorear podemos encontrar en que parte especifica encontramos un error.

Determinar Fallos

Una vez que el fallo es identificado, dependiendo de la meta de la auditoria debe ser necesario explotar los bugs, esto es típicamente un proceso manual que requiere conocimiento en seguridad. Esto puede ser llevado por otra persona que no haya estado desde el inicio.

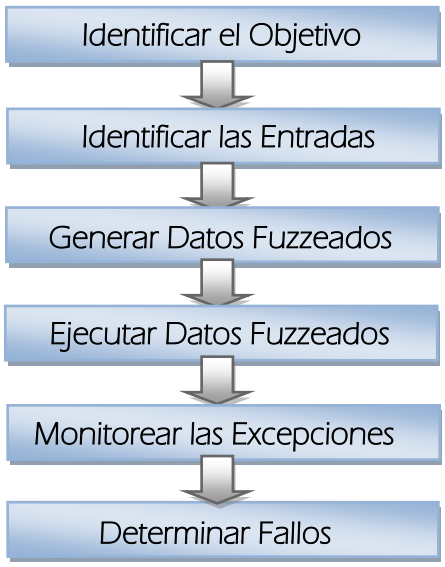


Figura 2.2 Fases del Fuzzing

2.4.2. Fases del Test de Penetración

Recopilación de información

En ella utilizaremos los buscadores, herramientas de análisis de DNS y demás herramientas para obtener información de nuestra víctima. Además, podemos hacer una exploración de metadatos de los documentos, imágenes y otros tipos de archivos que tengamos a nuestro alcance navegando.

Enumeración

En ella trataremos de conseguir direcciones IP de nuestra víctima, nombres de usuarios y contraseñas válidas de su entorno y nombres de servicios y aplicaciones accesibles, y todo aquello que luego nos pueda ayudar a lanzar nuestro ataque. Dejar claro que esta fase al igual que la primera, solo es de investigación, no se llevará a cabo ningún ataque, será más bien la realización de una lista de checks de elementos existentes.

Análisis

En ella empezaremos a actuar sobre los sistemas encontrados, los analizaremos en busca de vulnerabilidades, ya sea en la infraestructura, los sistemas operativos, los servicios disponibles o las aplicaciones existentes.

Explotación

En esta fase realizaremos la intrusión en el sistema y obtendremos evidencias de nuestra intrusión para la posterior documentación o la demostración de que hemos realizado la intrusión.

Documentación

En esta fase plasmaremos de forma entendible y accesible todos nuestros descubrimientos. Registraremos todo lo que hemos descubierto sobre los sistemas, realizaremos informes de nuestras intrusiones y las evidencias de estas, realizaremos una presentación concisa y resumida de resultados, y señalaremos aquellos puntos que creamos que requieren especial importancia o que provocan los problemas más graves o inmediatos.

2.5. Etapas del Fuzzing

El funcionamiento de los fuzzers suele componerse de las siguientes etapas:
Obtención de datos: dependiendo del tipo de fuzzing deseado y según la implementación de la herramienta, se obtendrán los datos a enviar de una lista estática almacenada en archivos, o del propio código fuente, o se generará en el momento según las configuraciones efectuadas. Este proceso sólo se puede realizar al inicio de la sesión o justo antes de cada envío.
Envío de datos: una vez que se dispone de la información que se desea enviar a la aplicación, se realizará el proceso, dependiendo, si se hace a través de una red informática o de si se quiere realizar un chequeo local.
Análisis: después de realizado el envío, sólo quedará esperar los resultados del fuzzing. Si no se espera ninguna respuesta por parte del objetivo, se deberá estar alerta por si se produce un comportamiento inesperado. Si, en cambio, se recibe una respuesta, entonces en este momento se comprobará si ésta indica un comportamiento normal o si, por el contrario, el ataque ha tenido éxito y la aplicación ha quedado inestable.

g

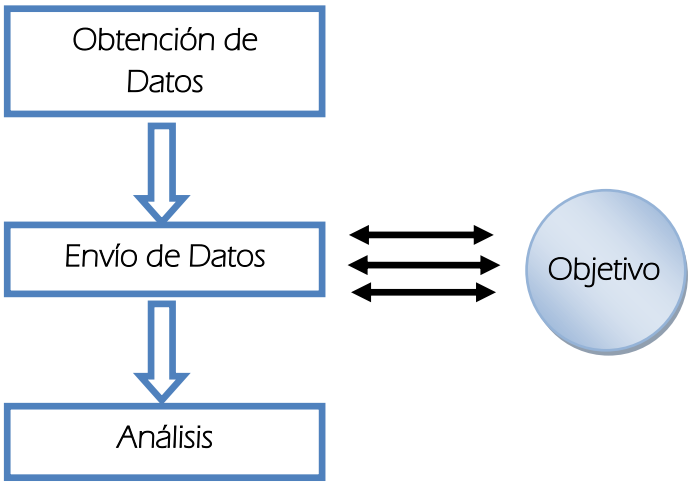


Figura 2.3 Arquitectura genérica de un fuzzer

En este proceso los puntos importantes son la obtención de los datos y el análisis posterior al envío. Ambos son importantes porque los datos deben ser útiles a la hora de crear un punto de inestabilidad en el objetivo y sin el análisis nunca se sabría si el ataque ha sido efectivo o no.

Gracias a esta técnica, y a los diversos fuzzers existentes, se han descubiertos miles de bugs, dejando claro que su efectividad está más que demostrada.

2.6. Representación de Datos

Los protocolos son necesarios para facilitar la comunicación. A veces se definen protocolos o normas. La información individual se prologa con metadatos.

Protocolos de Campos

Muchas decisiones deben hacerse en el diseño de un protocolo, pero una de las más importantes es cómo el protocolo se delimitará o dividirá en distintos componentes. El envío y recepción de las máquinas deben estar conscientes

de cómo interpretar los elementos individuales dentro de los datos, y esto es definido por el protocolo. Tres enfoques típicos para superar este reto, se fijan los campos de longitud, los campos de longitud variable o campos delimitados. Protocolos simples de texto plano son a menudo delimitados por caracteres. Ejemplos tales como:
XML, es un tipo de archivo que utiliza los campos delimitados por caracteres, pero en lugar de simplemente usar comas, utiliza caracteres simples para indicar el final de un campo, XML aprovecha los caracteres múltiples para delimitar el archivo. Los elementos se definen con los caracteres tanto apertura como de cierre de paréntesis angulares (< >).

Campos de longitud fija predefinen un número determinado de bytes a ser utilizados por cada campo. Este enfoque se utiliza comúnmente en las cabeceras de protocolos de red como Ethernet, protocolo de Internet (IP), Transmisión Control Protocolo (TCP) y Protocolo de datagramas de usuario (UDP) como cabeceras que requieren información estructurada cada cierto tiempo.

Protocolos de Texto plano

El término protocolo de texto plano se utiliza para referirse a un protocolo en el que convierto los datos en bytes, en su mayoría caen en el rango ASCII imprimibles. Esto incluye todos los números, letras mayúsculas y minúsculas, símbolos como signos de dólar y porcentaje, así como retornos de carro (\ r, 0x0D byte hexadecimal), las nuevas líneas (\ n, hexa bytes 0x0A), pestañas (t \, hexagonal bytes () X09), y los espacios.
Un protocolo de texto plano está diseñado para ser legible. Los protocolos de texto plano son generalmente menos eficientes que sus contrapartes binarias ya que ocupan más memoria, pero hay muchas situaciones en las que es deseable contar con un protocolo que sea legible. El canal de control del Protocolo de Transferencia de Archivos (FTP) es un ejemplo de un protocolo

de texto plano. El canal de transferencia de datos, en cambio, es capaz de transmitir datos binarios. FTP se utiliza para cargar y descargar datos a una máquina remota. El hecho de que el tráfico FTP de control es legible permite la comunicación, para ser manipulados manualmente usando herramientas de línea de comandos.

Protocolos Binarios

Los protocolos binarios son más difíciles para los seres humanos de descifrar, ya que en lugar de un flujo de texto legible es una secuencia de bytes en bruto. Sin una comprensión del protocolo, los paquetes no serán especialmente significativos.
Para el fuzzing, una comprensión de la estructura del protocolo es esencial si va a lugares de destino significativa dentro del protocolo. AIM (AOL Instant Messenger) es un ejemplo de un protocolo propietario. Aunque oficialmente no documentado, un montón de detalles acerca de la estructura del protocolo están disponibles gracias a los esfuerzos de ingeniería inversa de otros. El protocolo de AIM se conoce oficialmente como OSCAR (sistema abierto de comunicación en tiempo real).

Protocolos de Red

Tanto los protocolos FTP y AIM detallados anteriormente son ejemplos de protocolos de red. El Internet es todo acerca de protocolos de red y no hay escasez de ellos.
Tenemos protocolos para la transferencia de datos, enrutamiento de correo electrónico, streaming de medios de comunicación, la mensajería instantánea.

¿Cómo se desarrollaron los protocolos de red?

La respuesta a la pregunta depende en gran medida sobre si el protocolo es abierto o propietario. Protocolos propietarios pueden ser desarrollados por un grupo cerrado de una misma empresa para ser utilizado por productos específicos gestionada y controlada por la misma compañía. En cierto modo, los desarrolladores de protocolos propietarios tienen una ventaja inherente, ya que sólo necesita llegar a un consenso entre un pequeño grupo de desarrolladores momento del acuerdo sobre una norma. Por otra parte, los protocolos de Internet son intrínsecamente abiertos y por lo tanto requieren un consenso entre los diversos grupos. En general, los protocolos de Internet se han desarrollado y mantenido por la Internet Engineering Task Force (lETF). El IETF tiene un largo proceso para publicar y obtener información sobre los estándares de Internet propuesta, que comienza con la publicación de solicitudes de comentarios (RFC), que son documentos públicos. Tras un debate adecuado y la revisión, entonces puede RFC aprobado por la IETF como estándares de Internet.

2.7. Limitaciones

Fallas de Acceso de Controles

Algunas aplicaciones requieren capas de privilegios para soportar múltiples cuentas o niveles de usuarios, por ejemplo un sistema de calendario en línea puede ser accedido vía web, la aplicación necesita especificar un administrador que controla quien ha entrado al sistema, estos podrían ser un número especial de usuarios capaces de crear calendarios, otros usuario solo tendrán privilegio de lectura. La forma más básica permitirá asegurar que usuarios regulares o podrán realizar tareas administrativas.

Un fuzzer permitiría descubrir las fallas en el software calendario que le permitiría a un atacante tomar un completo control del sistema.

Diseño Lógico Pobre

Los Fuzzers no son las mejores herramientas para identificar un pobre diseño lógico. Consideremos este ejemplo, las vulnerabilidades descubiertas en Veritas Backup Exec permiten a los atacantes un remoto acceso a los Servidores Windows al momento de crear, modificar o eliminar claves de registros, esto podría comprometer completamente el sistema. Esta existe debido a la implementación de la interfaz de las llamadas remotas (RPC) en el Protocolo de Control de Transmisión TCP.

Puertas Traseras

Para un fuzzer con límites o sin información sobre la estructura de la aplicación objetivo, una puerta trasera no es vista diferente que otro objetivo lógico, como una pantalla de autentificación. Ambos son simples vectores de entrada recibiendo credenciales de autentificación. Además, a menos que el fuzzer tenga la suficiente información para reconocer autentificaciones exitosas, no hay manera para identificar la autentificación exitosa usando un código de contraseña difícil, ya que esto es aleatorio.

Corrupción de Memoria

La corrupción de memoria frecuentemente provoca fallas al proceso objetivo. Este tipo de problemas puede ser reconocido por algunos síntomas como una denegación de servicios. Sin embargo, algunos problemas de corrupción de memoria son exitosamente manejados por la aplicación objetivo y nunca podrían ser reconocidos por un simple fuzzer.

Consideremos por ejemplo, un formato de caracteres vulnerable que podría pasar sin detectar adjuntos con un debugger al proceso objetivo. Un formato de caracteres vulnerable frecuentemente puede afectar a la maquina a nivel de código. Por ejemplo en una maquina Linux x86, un síntoma de un ataque

de formato de caracteres puede incluir un %n como en la siguiente instrucción:

```
mov %ecx,(%eax)
```

Si el fuzzer está usando datos aleatorios al proceso con el mismo formato de caracteres como %n, el registro podría no contener en algunos casos direcciones escritas pero crearía palabras basuras en la pila, una violación de señales de segmentación, esto terminaría el proceso y comenzaría uno nuevo, lo cual podría permitir ejecuciones continuas dentro de un proceso sin restaurarlo.

Vulnerabilidades Multifase

La explotación no siempre es tan simple como atacar simples debilidades. Complejos ataques frecuentemente envuelven varias vulnerabilidades que comprometen un maquina. Fuzzing podría ser usado para identificar fallas individuales pero generalmente no evalúa todos estos cambios juntos, como una serie de vulnerabilidades o otros eventos inesperados para identificarlos como un multivector de ataque.

2.8. Tipos de Fuzzers

Hemos visto diferentes métodos y aproximaciones al fuzzing, ahora veremos algunos tipos específicos del fuzzers. Estos están basados en los diferentes tipos de objetivos.

2.8.1. Fuzzers Locales

En el mundo UNIX, el setuid de aplicación es permitir a un usuario normal que temporalmente gane algunos privilegios, esto hace algunas aplicaciones un objetivo obvio para el fuzzing. Haciendo vulnerable a las

aplicaciones ya que pueden permitir a los usuarios escalar privilegios y ejecutar código según escoja.

Aquí se puede tener dos diferentes objetivos:

El primero la línea de comandos fuzzing se enfoca en pasar argumentos malformados al setuid de la aplicación en línea de comandos. El segundo también pasaría argumentos malformados pero en diferente forma, en este caso es pasado a través del ambiente shell de UNIX.

Línea de Comandos Fuzzers

Cuando las aplicaciones comienzan, frecuentemente se envían argumentos pasados por el usuario, a continuación una simple forma de parámetros para desbordamiento de pila (stack overflow) en línea de comandos:

```
#include <string.h>
int main (int argc, cha **argv)
{
   char buffer [10]:
   strcpy (buffer, argv[1]):
}
```

Variables de Ambiente Fuzzers

Otro tipo envuelve los vectores de variables de ambiente, considerando el ejemplo anterior tenemos los valores inseguros desde el ambiente de usuario:

```
#include <string.h>
int main (int argc, cha **argv)
{
   char buffer [10]:
   strcpy (buffer, getenv(“HOME”)):
}
```

Aquí hay algunas formas efectivas en el uso de variables de ambiente en la aplicación pero no hay muchas herramientas para la automatización a pesar de que es un simple proceso. Muchos buscadores de seguridad han realizado sus propios scripts para estas tareas, esta puede ser una de las razones por la que no se encuentran herramientas.

Formato de Archivos Fuzzing

Muchas aplicaciones (cliente, servidor), deben de tener un mismo trato para los archivos de entrada y salida, por ejemplo los gateways antivirus necesitan convertir los archivos comprimidos para el diagnostico, esta aplicación puede ser vulnerable a que ocurra una conversión maliciosas de estos archivos.

Aquí es donde aparece el formato para archivos fuzzing. Un formato de archivo especial que crea diferentes archivos malformados que luego son ejecutados usando la aplicación. Aunque los métodos específicos usados en los archivos no son exactamente los mismos que otros tipos de fuzzing, la idea general es la misma.
Los formatos frecuentes son:

- FileFuzz de Michael Sutton, interfaz GUI Windows, basada en herramientas de archivos fuzzing.
- notSPIKEfile y SPIKEfile por Adam Greene, herramienta basada en UNIX.

2.8.2. Fuzzers Remotos

Es el software objetivo que se usa en una interfaz de red. Las aplicaciones habilitadas en la red son comúnmente el objetivo de este tipo de fuzzer, con el crecimiento de Internet, virtualmente todas las corporaciones ahora

tienen servidores públicos para sus páginas web, e-mail, DNS resoluciones, y más. Una vulnerabilidad en alguno de estos sistemas puede proporcionar un ataque con acceso a los datos y comprometer los servidores.

Protocolos Fuzzers Network

Pueden dividirse en dos categorías: aquellos que tienen por objetivo simples protocolos y los que tienen por objetivo protocolos complejos. Se enlistaran las características comunes para cada tipo.

Protocolos Simples

Frecuentemente tienen una simple autentificación o no son autentificados del todo. Ellos están basados en texto ASCII. Adicionalmente no manejan estados dentro de la aplicación, por ejemplo un ejemplo de protocolo simple es FTP, en este, todos los controles de canales de comunicación están en texto plano ASCII, para la autentificación solo necesitamos un texto plano con un usuario y contraseña.

Protocolos Complejos

Son típicamente comprimidos de datos binarios, ocasionalmente contienen una cadena ASCII, para la autentificación puede requerir encriptación o alguna de autentificación más compleja.

Un ejemplo de esto es el protocolo Microsoft Remote Procedure Call (MSRPC), un protocolo binario que requiere varios pasos para establecer comunicación con un canal antes de la transmisión de datos. Este requiere la descripción de la longitud de campos y fragmentación. Sin embargo ni es un protocolo fácil de implementar para el fuzzing. Se usan las siguientes herramientas para esto:

- SPIKE de Dave Aited, primer framework para fuzzing público, esto incluye scripts de fuzzing pre-generados para varios protocolos populares.
- Peach de Michael Eddington, una plataforma para framework fuzzing escrita en Python, es flexible y puede ser usada para una red objetivo.

Web Application Fuzzers

Son populares, una conveniente vía para el acceso de servicios back – end como e-mail y compras online. Lo primero que busca son las vulnerabilidades únicas para cada aplicación como SQL injection, Cross Site Scripting (XSS), entre otras.
Se necesita una comunicación vía HTTP y capturar las respuestas de algunos análisis para la identificación de existencia de vulnerabilidades.
Entre las herramientas más usadas tenemos:

- WebScarab de OWASP, una aplicación auditora Open Source con capacidades de fuzzing.
- SPI Fuzzer por SPI Dynamics, una aplicación comercial HTTP que incluye el scanner de vulnerabilidad WebInspect.
- Codenomicon, test suite HTTP.

Buscadores Web Fuzzers

Estos son técnicamente solo un tipo especial de formato de archivo Fuzzers, solo son clases populares aplicaciones basadas en web. Estos frecuentemente utilizan funcionalidades en HTML para automatizar el proceso de fuzzing.

Por ejemplo, la utilidad Mangleme de Icamtuf, es una de las primeras herramientas disponibles para los buscadores fuzzing, utiliza el tag <META REFRESH> para cargar casos de pruebas en modo automático. La única

característica de los buscadores web es que permiten una completa automatización del lado del cliente fuzzer sin ningún contenedor complejo alrededor de la aplicación.

Las herramientas más utilizadas son:

- Mangleme de Icamtuf, el primer fuzzer HTML, este es un script CGI que envía datos HTML al buscador.
- DOM-Hamoi y Hamachi de H.D .Moore y Aviv Raff, un fuzzer DHTML.

2.8.3. Fuzzers en Memoria

Algunas veces durante las pruebas, ciertos obstáculos impiden un rápido y eficiente fuzzing. Esto es cuando los Fuzzers en memoria deben ser usados. Una implementación envuelve una toma instantánea de un proceso y una rápida carga de datos defectuosos en una de las rutinas de entrada. Esto tienes sus ventajas y desventajas.

Entre las ventajas tenemos:

- Velocidad: No requieren el ancho de banda de la red, pero necesitan correr códigos recibiendo paquetes fuera de la red y estos pueden ser eliminados.
- Atajos: Algunas veces los protocolos usan encriptación personalizada, compresión de algoritmos o validación de códigos de checksum.

Entre las desventajas tenemos:

- Falsos Positivos: los datos enviados pueden instanciarse de alguna manera que nunca serán enviados lo cual da lugar a falsos positivos.

- Complejidad: El método a usar puede ser demasiado difícil de implementar.

2.9. Metodologías para el Descubrimiento de Vulnerabilidades

Caja Blanca

Se realiza basándose en el conocimiento de cómo el sistema se aplica, incluye el análisis de flujo, control de flujo, el flujo de información, prácticas de codificación y manejo de errores dentro del sistema, para probar el comportamiento del software. Estas pruebas se pueden realizar para validar si la implementación del código va acorde con el diseño, para validar la funcionalidad de seguridad y descubrir vulnerabilidades explotables.

Estas pruebas requieren acceso al código fuente, se pueden realizar en cualquier momento del ciclo de vida del software. Es una buena práctica realizar este método durante la fase de pruebas unitarias.

Las pruebas de caja blanca requieren el conocimiento de lo que hace a un software seguro o inseguro, cómo piensa un atacante y cómo usar las diferentes técnicas y herramientas. El primer paso es comprender y analizar la documentación del diseño, código fuente. El segundo paso es crear pruebas que se aprovechen del software. El tercer paso es realizar las pruebas de eficacia.

Ventajas

- Cobertura: ya que debido a que todo el código fuente está disponible, una revisión del código permite una cobertura completa. Todas las posibles rutas del código pueden ser auditadas para encontrar vulnerabilidades. Esto puede dar lugar a falsos positivos y las rutas del código no podrían ser accesibles durante la ejecución.

Desventajas

- Complejidad: las herramientas de análisis de código fuente son imperfectas y producen falsos positivos. El reporte errores producidos por las herramientas deben ser revisadas por programadores para identificar problemas o alguna vulnerabilidad en la seguridad.
- Disponibilidad: El código fuente no siempre está disponible

Caja Negra

Es un método que pone a prueba la funcionalidad de una aplicación. No se requiere conocimientos específicos del código de la aplicación. Estos casos se construyen de acuerdo a los requerimientos y especificaciones. Estas pruebas pueden ser funcionales o no funcionales. El que prueba selecciona las entradas válidas e inválidas y determina la salida correcta.

Puede ser aplicada a todos los niveles de pruebas de software.

Ventajas

- Disponibilidad: Es siempre aplicable.

Caja Gris

Combina las técnicas de caja blanca con las pruebas de entrada de la caja negra. Explora las rutas que son directamente accesibles desde las entradas del usuario o interfaces externas al software.

Ventajas

- Cobertura: El análisis de esta prueba puede ser utilizada para mejorar las técnicas de caja negra.

2.9.1. Métodos Usados

El fuzzer trataría combinaciones de ataques a:
-Números (enteros con / sin signo / float).
- Caracteres (urls, entradas de línea de comandos).
- Metadatos: texto por el usuario de entrada (la etiqueta ID3).
- Secuencias binarias.
Un enfoque común para el fuzzing es definir las listas de “los valores conocidos-a-ser-peligroso” (vectores fuzz) para cada tipo.

1. Mutación o Fuerza Bruta: Es un método efectivo que consiste en partir de una entrada válida y realizar ciertas mutaciones de esos datos, con la intención de que sigan siendo válidas para la aplicación pero lo suficientemente inesperadas para lograr el objetivo.

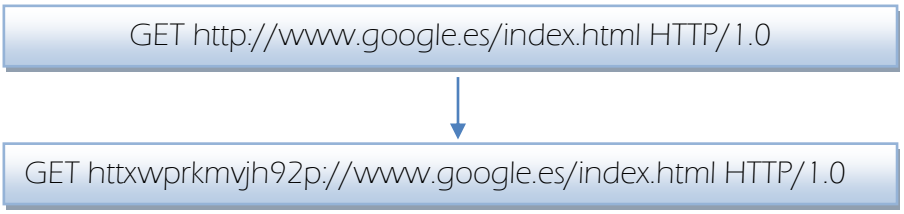


Figura 2.4 Mutación de Datos

2. Generación: Se trata de un proceso más lento que el anterior, ya que primero se deben crear los datos antes de enviarlos, pero puede descubrir fallos que el otro método obviaba.
Normalmente, los datos obtenidos también se sustituirán en una entrada válida del objetivo, ya que si se envía simplemente un dato aleatorio, se corre el riesgo de que el programa lo deseche sin ni si quiera procesarlo.

Dependiendo de la forma de generar los datos se encuentran otros 2 subgrupos. Los recursivos, que se obtienen de la iteración sobre un alfabeto dado o de la repetición de un mismo carácter:

Permutación: Teniendo en cuenta, por ejemplo, el alfabeto hexadecimal "0,1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F" y buscando datos de 6 caracteres se podrían obtener 16^6 combinaciones posibles para sustituir de la forma siguiente:

Figura 2.5 Permutación de datos

Repetición: se usa generalmente para la búsqueda de fallos del tipo buffer overflow. Se suele usar un incremento en cada repetición para no alargar el proceso inútilmente, ya que los puntos críticos se encuentran cerca de los límites de potencias de 2, como 256, 512, 1024, etc. Como se muestra en la Figura 4.

El otro subgrupo es el que hace uso de la sustitución, que se centrará en reemplazar cierta parte de una entrada válida por los elementos de un vector de fuzzing, que es una lista de posibles test a realizar, dependientes de la vulnerabilidad a chequear. Así pues, dos de los casos más importantes son los siguientes:

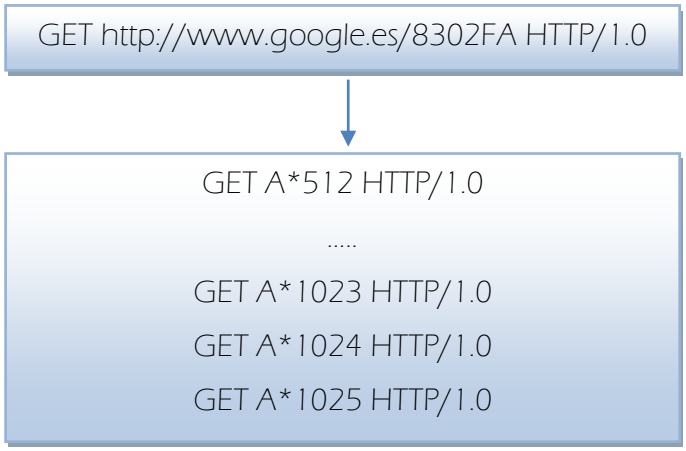


Figura 2.6 Incremento del número de caracteres

Integer overflow: ocurre cuando un programa recibe en una entrada un número que está fuera del rango que es capaz de manejar: un cero, un número negativo, un número mayor o menor que el rango actual, o uno con diferente tipo numérico (decimal en vez de entero). Este agujero puede derivar en un ataque de buffer overflow. Un posible ejemplo de esto puede ser la modificación del campo Content-Length en una petición HTTP.

Errores de cadenas de formato (format string): este tipo de ataques se producen cuando se introducen ciertos elementos para proporcionar formato a las cadenas de texto y no existe un mínimo control de seguridad. De esta forma, se puede conseguir una denegación de servicio o incluso la ejecución de código arbitrario en el sistema.
Para detectar vulnerabilidades en aplicaciones con técnicas de fuzzing podemos utilizar Fusil una biblioteca escrita en Python que sirve para crear programas de fuzzing.

Uno de los mejores métodos para prevenir la aparición de bugs de seguridad en aplicaciones en producción es mejorar el Ciclo de Vida de

Desarrollo del Software (en inglés Software Development Life Cycle, o SDLC), incluyendo la seguridad.

3. ANALISIS DE REQUERIMIENTOS Y DISEÑO

3.1. Requerimientos Funcionales

Desde el punto de vista de los requisitos funcionales de seguridad, las normas aplicables, las políticas y reglamentos conducen ambas a la necesidad de un tipo de control de seguridad, así como el control de la funcionalidad. Estos requerimientos también son referidos como "requerimientos positivos", ya que se espera que la funcionalidad pueda ser validada a través de pruebas de seguridad. Ejemplos de requerimientos positivos son los siguientes: "la aplicación se bloqueará al usuario después de seis intentos de inicio de sesión fallidos" o "las contraseñas deben ser de seis caracteres alfanuméricos como mínimo". La validación de los requerimientos positivos consiste en constatar la funcionalidad esperada y, como tal, pueden ser probados re-creando las condiciones de prueba y ejecutando las pruebas de acuerdo a las entradas predefinidas y constatando los resultados previstos como la condición de no pasa/si pasa.

Ejemplos de alto nivel de diseño de requerimientos de seguridad para la autenticación pueden ser:

- Proteger las credenciales de usuarios y secretos compartidos, en tránsito y almacenadas
- Enmascarar datos confidenciales cuando se visualicen (por ejemplo, contraseñas, cuentas)
- Bloquear la cuenta del usuario después de cierto número de intentos de acceso fallidos
- No mostrar al usuario errores específicos de validación como resultado de un acceso fallido
- Solamente permitir contraseñas alfanuméricas, que incluyan caracteres especiales y que tengan seis caracteres mínimos de longitud, todo esto para limitar ataques desde la interface

- Permitir la funcionalidad de cambio de contraseña únicamente a usuarios autenticados validando la antigua contraseña, la nueva contraseña y la respuesta a la pregunta de seguridad, esto para evitar ataques de fuerza bruta a la contraseña a través de la funcionalidad de cambio de contraseña.
- El formulario de re-inicialización de contraseña debería validar el identificador del usuario y su correo electrónico registrado antes de enviar la contraseña temporal al usuario. La contraseña temporal generada debería ser una contraseña de un solo uso (One Time Password).

3.2. Requerimientos No Funcionales

Los requerimientos no funcionales influyen en la operatividad del software. Para el desarrollo de la herramienta consideramos los siguientes:

1. **Reproducibilidad:** Un requerimiento necesario para una herramienta de fuzzing es que ésta reproduzca los resultados de ambas pruebas, las secuenciales y las individuales. Al crear este tipo de herramientas fuzzer, se debe de proveer este con una lista de números de casos de pruebas maliciosas. Se deben de generar reportes con la finalidad de informar al usuario los fallos de la aplicación

2. **Documentación:** La Documentación de los varios resultados de las pruebas son también de gran utilidad ya que es difícil reunirse con el creador del fuzzing, es más factible leer la documentación en caso de alguna duda.

3. **Reusabilidad:** Si queremos usar la herramienta fuzzer para un tipo de formato de archivo diferente, no deberíamos tener que reescribir de nuevo

el código más bien se debería de crear un componente reusable en el cual podamos cambiar lo que queremos a nuestras necesidades y así minimizaríamos el tiempo para crear nuestra propia herramienta.

3.3. Diseño

3.3.1. Arquitectura

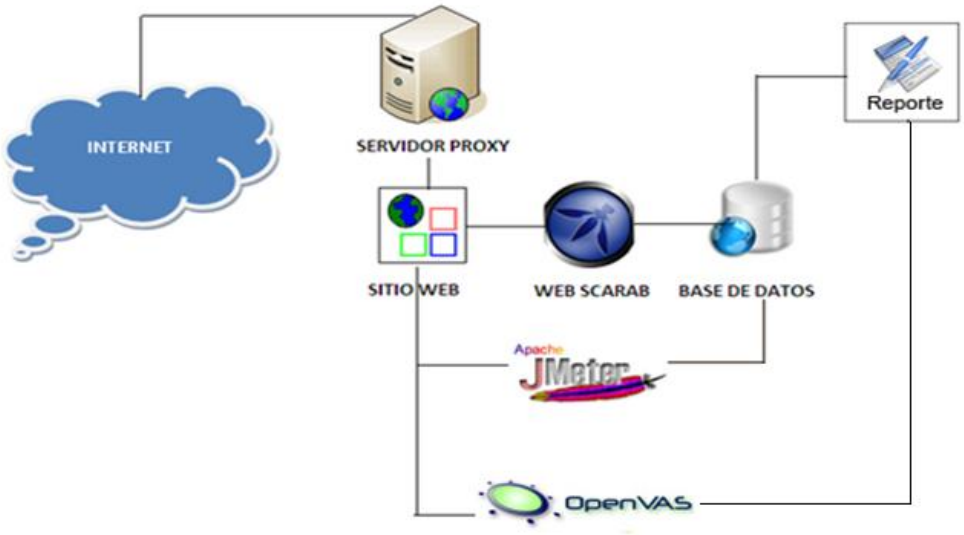


Figura 3.1 Diseño del Producto

El Web Fuzzing se centra específicamente en los paquetes que se ajustan a la especificación de HTTP. Se registra todas las comunicaciones entre el usuario y el servidor web, esto incluye todas las imágenes, archivos CSS, Javascript, parámetros, etc. De acuerdo al diseño y a las herramientas usadas, se podrá ofrecer un amplio servicio en el cual se podrá actuar como proxy HTTP, se interceptarán las peticiones HTTP request y HTTP response para poder estar informados de lo que hace el sitio web y observar los tipos de fallos o atentados que se podrían producir.

Se interceptará tanto el nombre del usuario “hacker” y la contraseña. Un proxy HTTP es capaz de ver la contraseña a pesar de que cada carácter se

sustituye por un asterisco en la aplicación. Con un proxy HTTP que puede manipular cualquier solicitud / respuesta no sólo el proceso de inicio de sesión. La entrada de nombre de usuario es vulnerable a inyección SQL para que podamos tratar con el fuzzing de encontrar otras posibles inyecciones

Se captará el proceso de inicio de sesión en la ficha de resumen. Una vez que esto ha sucedido se encontrará la conversación entrada en la ficha de resumen. Después de haber encontrado la conversación se la puede usar a manera de plantilla y esta enviará los parámetros y las cabeceras asociadas a esta solicitud / conversación a la pestaña de fuzzing.

Se podría añadir parámetros a la petición y ver cómo reacciona la aplicación web. Una vez que el fuzzing haya terminado de hacer todas las peticiones, una vista de los resultados se mostrará para comprobar si algunos de los parámetros enviados como ataque tuvieron éxito.

Se analizará el trafico de acuerdo a la cantidad de usuarios que quieran acceder a un sitio web y así podemos analizar el retraso que existe entre cada petición, el número de usuarios que accedieron sin inconveniente y la demora que existe.

3.3.2. Características

Se provee de un número de plugins, cuyo objetivo principal, por el momento, es agregar funcionalidad de seguridad, reportar las vulnerabilidades y explotarlas. Estos plugins incluyen:

- Proxy - Observa el trafico entre el navegador y el servidor Web. Es capaz de observar tanto HTTP como trafico HTTPS cifrado. Permite al operador controlar las peticiones y respuestas que pasan por el proxy.
- Intercepción Manual - Permite al usuario modificar peticiones y respuestas HTTP y HTTPS "al vuelo", antes de que ellas alcancen el servidor o el navegador.
- Simulador de ancho de banda - permite al usuario emular una red más lenta, de manera que observe como se desempeña su sitio cuando es accedido, se observa el retraso y la cantidad en milisegundos en que se demora cada usuario en acceder a dicho sitio.
- Peticiones manuales - permite editar y reenviar peticiones anteriores la creación de peticiones nuevas completas.
- Análisis de identificadores de sesión - recolecta y analiza un número de cookies (y eventualmente parámetros en el URL también) para determinar visualmente el grado de aleatoriedad y predictibilidad.
- XSS/CRLF – se buscan datos controlados por el usuario en los encabezados y cuerpo de las respuestas HTTP para identificar posibles inyecciones CRLF (partición de respuesta HTTP) y vulnerabilidades de secuencia de comandos en sitios cruzados (XSS).

3.4. Herramientas de Desarrollo

Las herramientas que han sido seleccionadas para el desarrollo del proyecto fueron elegidas por diferentes criterios como la experiencia en el manejo de la herramienta, la curva de aprendizaje de nuevas herramientas adoptadas y la facilidad de implementación.

3.4.1. Herramientas de Software

JRE (Java Runtime Environment): Es un conjunto de utilidades que permite la ejecución de programas Java.

En su forma más simple, el entorno en tiempo de ejecución de Java está conformado por una Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.

Un usuario sólo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JDK, que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java.

NetBeans: Es una plataforma que permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones

construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos.

WebScarab: Es una herramienta multiplataforma (Java) que sirve para realizar pruebas de aplicaciones web funcionando como proxy interceptor:

- Registra todos los accesos (documentación).
- Modificación arbitraria de peticiones y respuestas.
- Extensible a través de complementos: análisis de identificadores de sesión, pruebas automáticas de parámetros (*fuzzer*).

JMeter : Es un proyecto de Apache Jakarta que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web.
JMeter puede ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, Servicios web, JMS, HTTP y conexiones TCP genéricas. JMeter puede también ser configurado como un monitor, aunque es comúnmente considerado una solución ad-hoc respecto de soluciones avanzadas de monitoreo.

OpenVas: Es el acrónimo de *Open Vulnerability Assessment System*, un completo escanner de vulnerabilidades que permite evaluar los riesgos de seguridad en los equipos de una red y cerrar sus vulnerabilidades.

3.5. Los ataques de seguridad más comunes

El *spyware* y los *keyloggers* se han colocado como los ataques más comunes en compañías que sufrieron las brechas de seguridad más severas en Estados Unidos.

Un reporte titulado “Anatomía de las Brechas de Información 2009” elaborado por Verizon Business dio a conocer las formas más típicas en las cuales los cibercriminales están afectando de manera importante a las compañías. La información fue extraída de su servicio de cómputo forense.
El estudio muestra cómo identificar las señales de alerta, cómo operan los ataques, cómo ingresan los intrusos a los sistemas, detrás de cuál información va y qué industrias son típicamente afectadas.

Los 15 ataques más comunes identificados son:

Keyloggers y Spyware: Dichos ataques permiten instalarse silenciosamente en la PC con el fin de enviar datos sobre la información que la víctima escribe o almacena en el sistema, incluso, sobre sus hábitos en Internet. Por lo general se instalan en sistemas de usuarios finales y servidores.
Impacto (pérdida de datos): en número de casos el factor es del 82% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 19%.

Backdoor o puerta trasera: Estas herramientas dan acceso remoto para controlar los sistemas infectados y, cuando son ejecutados, corren encubiertamente. Por lo general, se instalan en los servidores, el objetivo son aquellos procesos que almacenan o transmiten datos sensibles y le da una ventaja para el atacante
Impacto (pérdida de datos): en número de casos el factor es del 79% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 18%.

Inyección SQL: Es una técnica de ataque utilizada para explotar vulnerabilidades en páginas Web que tienen una ruta de comunicación con bases de datos. El blanco de ataques de inyección SQL son servidores de bases, en especial los que almacenan datos sensibles o se encuentran en un

entorno de red que contiene los datos sensibles. La inyección de código SQL se utiliza para todo tipo de transacción de datos, pero es más asociado con los datos de tarjetas de pago.
Impacto (pérdida de datos): en número de casos el factor es del 79% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 18%.

Abuso al sistema vía acceso privilegiado: Es el abuso deliberado de recursos, accesos o privilegios concedidos a una persona por una organización. Cualquiera y todos los activos pueden estar comprometidos. A menudo los objetivos son las direcciones IP y tipo de información corporativa.
Impacto (pérdida de datos): en número de casos el factor es del 1% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 17%.

Acceso no autorizado con credenciales predeterminadas: Son los métodos a través de los cuales los atacantes obtienen acceso a un dispositivo o sistema protegido con contraseñas y nombres de usuario predeterminados o estandarizados.
Casi siempre los objetivos son las aplicaciones, servidores y dispositivos de red debido a su compromiso con información personal.
Impacto (pérdida de datos): en número de casos el factor es del 1% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 17%.

Violación de usos aceptables y otras políticas: En realidad, este ataque no distingue si fue cometido de manera accidental o premeditada, la violación a una política fue tener graves consecuencias. Por lo general involucra directamente a los sistemas de usuario final y afecta a una gran variedad de activos y datos.

Impacto (pérdida de datos): en número de casos el factor es menor del 1% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 12%.

Acceso no autorizado mediante listas de control de accesos débiles o mal configuradas (ACL): Cuando hay las condiciones el atacante puede acceder a recursos y llevar a cabo acciones sin que la víctima se dé por enterada. Se relaciona con todo tipo de activos y datos, excepto formularios en línea, pero más comúnmente afecta a los dispositivos de red, aplicaciones y servidores.
Impacto (pérdida de datos): en número de casos el factor es del 66% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 10%.

Sniffers: Estas herramientas monitorean y capturan información a través de una red. Casi siempre se instala en los servidores, el objetivo son los procesos que almacenan o transmiten grandes cantidades de datos sensibles o se encuentran en un entorno de red que contiene los datos sensibles. Los paquetes sniffers pueden capturar cualquier tipo de datos, son responsables de la mayor parte de los datos comprometidos de las tarjetas de crédito.
Impacto (pérdida de datos): en número de casos el factor es del 89% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 9%.

Acceso no autorizado vía credenciales robadas: Para llegar a este punto, el atacante se valió de otros métodos para ganar acceso válido a sistemas protegidos sin ser detectado. Casi siempre sus objetivos son las aplicaciones, los dispositivos de red y servidores.
Impacto (pérdida de datos): en número de casos el factor es menor del 1% de los registros comprometidos.

Frecuencia: el número de casos de intrusiones es del 8%.

Pretexting: Con esta técnica el atacante crea una situación para manipular las creencias o sentimientos de la víctima y persuadirla de llevar a cabo una acción, como facilitarle información confidencial. El objetivo es el acceso a los activos de información. Por lo tanto, los empleados con mayores niveles de acceso o de responsabilidades dentro de la organización están comprometidos a este tipo de ataque.
Impacto (pérdida de datos): en número de casos el factor es del 2% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 8%.

Evasión de los procedimientos de autenticación: Las técnicas para evitar o evadir los mecanismos estandarizados de autenticación con el fin de obtener acceso no autorizado a un sistema. Casi siempre sus objetivos son las aplicaciones, los dispositivos de red y servidores.
Impacto (pérdida de datos): en número de casos el factor es menor del 1% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 6%.

Robo físico de un valor: Las brechas de información podrían comenzar con el robo de una laptop o incluso un servidor o una PC de escritorio. Es Más común en los activos móviles (es decir, ordenadores portátiles) y los datos en línea (es decir, los medios de comunicación, portátil).
Impacto (pérdida de datos): en número de casos el factor es del 2% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 6%.

Ataque de fuerza bruta: Es un proceso que requiere un alto nivel de automatización para intentar adivinar múltiples combinaciones de usuario y

contraseña para obtener acceso a un sistema. Casi siempre sus objetivos son las aplicaciones, los dispositivos de red y servidores.
Impacto (pérdida de datos): en número de casos el factor es del 7% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 4%.

RAM scraper: Una nueva forma de diseño de malware para capturar información en la memoria RAM. Casi siempre se instala en los servidores que procesan, almacenan o transmiten datos de tarjetas de pago.
Impacto (pérdida de datos): en número de casos el factor es del 1% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 4%.

Phishing y sus variantes: Una técnica de ingeniería social en la cual un atacante utiliza comunicaciones electrónicas fraudulentas para provocar que el destinatario facilite información. el objetivo es el acceso a los activos de información o las cuentas externas. la información personal son los activos que están más en riesgo.
Impacto (pérdida de datos): en número de casos el factor es del 4% de los registros comprometidos.
Frecuencia: el número de casos de intrusiones es del 4%.

4. PLAN DE NEGOCIOS

4.1. Plan Estratégico

Visión

El Test de Penetración se convertirá en un producto con un enfoque en el mercado ecuatoriano, necesario para la validación y prueba de software en empresas medianas y grandes, entregando la más alta calidad y un servicio único.

Misión

Ser pioneros en las herramientas de seguridad informática basadas en aplicaciones fuzzing.

4.1.1. Análisis Externo de la Empresa

Oportunidades

- Pocas empresas en desarrollo de productos enfocados en Test de Penetración.
- Empresas enfocadas en seguridad informática no tienen paquetes definidos para Test de Penetración.
- Empresas en el mercado local necesitan productos que validen sus aplicaciones.
- El país se encuentra en un auge tecnológico, abierto a las nuevas tecnologías, en desarrollo.

Amenazas

- Consumidores no interesados en el producto.
- Empresas locales no entiendan el porqué del producto.
- Poca demanda del producto debido a desinformación.

- Poco conocimiento de las herramientas para los test de penetración.
- Barreras comerciales

4.1.2. Análisis Interno de la Empresa

Fortalezas

- Nos encontramos actualmente desarrollando el producto para lanzar una versión beta, las personas involucradas en esto tienen total disponibilidad.
- Nuestra visión hacia el futuro es posicionar el producto como una herramienta necesaria e indispensable para los consumidores.
- El mercado el cual vamos a llegar es nuevo por lo tanto debemos crear la necesidad a los consumidores, ser innovadores.

Debilidades

- Poco conocimiento sobre lenguaje en que se va a desarrollar la aplicación.
- Falta de documentación en proyectos a utilizarse.
- Poco conocimiento de las herramientas para el desarrollo del software.
- Falta de experiencia en elaboración de plan de negocios.
- Falta de experiencia en marketing especializado.

4.1.3. Estrategias

Matriz FODA

	Fortalezas (F) 1. Disponibilidad 2. Posicionamiento del producto como una herramienta necesaria 3. Mercado nuevo	Debilidades (D) 1. Poco conocimiento sobre plataforma. 2. Falta de documentación en proyectos a utilizarse. 3. Falta de experiencia en elaboración de plan de negocios 4. Falta de experiencia en marketing especializado
Oportunidades (O) 1. Poca competencia 2. No hay herramientas de test de penetración definidas 3. Empresas en el mercado local necesitan productos que validen sus aplicaciones 4. Desarrollo	F2-O2 F2-O3 F3-O3 F1-O2	O3-D3 O3-O4 O4-D1

tecnológico		
Amenazas (A) 1. Consumidores no interesados en el producto 2. Empresas locales no entiendan el porqué del producto 3. Poca demanda del producto debido a desinformación 4. Poco conocimiento de las herramientas para los test de penetración 5. Barreras comerciales	F2-A1 F2-A2 F2-A3 F3-A5	D1-A4 D3-A1 D3-A2 D3-A3 D4-A1 D4-A2 D4-A3

Tabla 4.1 Matriz FODA

F2O2

Debemos aprovechar que en el mercado hay espacio para la inversión tecnológica en las entidades privadas y públicas.

F2O3, F3O3

Ofrecer información a las empresas sobre la importancia de la seguridad de la información desde el momento en que las aplicaciones son instaladas.

O3D3, O3D4

Los miembros del grupo de trabajo buscarán consultar y capacitarse apropiadamente. Al ser un mercado nuevo hay desinformación, por lo cual se buscaran nuevas oportunidades.

F2-A1, F2-A2, F2-A3, F3-A5

Al ser un producto innovador se creará la necesidad de su uso, realizando demostraciones y pruebas en empresas interesadas, a las cuales se les proporcionara información sobre ventajas y beneficios que obtendrán al usar el producto.

4.2. Delimitaciones del Proyecto

Se contará con un capital inicial aportado por nosotros como creadores y distribuidores del servicio, el cual servirá para los gastos de publicidad y comercialización.

La inversión inicial del proyecto comprende solo los gastos de publicidad y comercialización del servicio, de manera particular se asumirán los gastos por el empaquetamiento del software, ya que estamos usando herramientas Open Source, lo cual nos reduce el costo de inversión.

4.2.1. Competencia

En la actualidad en el mercado ecuatoriano hemos investigado a posibles competidores pero estos se encuentran en la región Sierra, en la Costa no se han encontrado competidores semejantes.

Hemos encontrado productos sustitutos, empresas que ofrecen servicios de auditoría informática, las cuales se encargan de la instalación y configuración de sistemas, mantenimiento, migración y seguridad de datos, permisos, todo esto ofrecido como un paquete de servicios.

4.2.2. Mercado Objetivo

Gobierno, instituciones públicas y privadas.

4.2.3. Factores Claves de Éxito

Los factores que se perciben como claves para el éxito en el sector son:

• Integrar los servicios que se presten dentro de la cultura y funcionamiento diario de la empresa-cliente. Estos no deben sentirse como un elemento lejano o sin importancia, sino como una parte funcional e importante de la empresa.

• Servicios complementarios. Servicio de atención de incidencias, asesorar a los trabajadores de la empresa-cliente en el uso de las nuevas tecnologías implementadas en su empresa.

• Red de contactos amplia y con garantía de calidad para la externalización de servicios. Esta red es necesaria para cumplir los plazos comprometidos y con la calidad ofertada.

4.3. Estudio de Mercado

4.3.1. Análisis de la Demanda

Nuestra demanda se basa en servicios públicos o servicios prestados con esto nos referimos a que el servicio es prestado hacia la comunidad en general, aunque no necesariamente por parte del estado, en nuestro caso nuestro sector está delimitado al área informática, actualmente conocidos como TIC (*Nuevas Tecnologías de la Información y de la Comunicación* o **IT** para *«Information Technology»*)

Tomando en cuenta que ofreceremos un servicio, la demanda es inelástica, es decir ante las variaciones del precio la cantidad demandada varía (en porcentaje) menos que la del precio.

En el mercado local no tenemos un competidor directo, ya que los que existen son variaciones de Hacking Ético pero ninguno ofrece este servicio con herramientas fuzzing.

El mercado objetivo está en crecimiento con lo cual nos aseguramos que tendremos clientes que necesitaran nuestros servicios, lo cual será a través de un contrato, por servicios prestados con nuestros potenciales clientes.

4.3.2. Análisis de la Oferta

En el Ecuador no tenemos datos oficiales sobre uso de nuevas tecnologías ni sobre herramientas enfocadas en Seguridad Informática.

4.3.3. Procedimientos y Controles de Calidad

La formación, la experiencia y la relación con el cliente durante el proceso de prestación del servicio son las principales garantías para la calidad del producto final.

Dada la importancia de la satisfacción final del cliente en un sector como el de los servicios informáticos, donde los clientes son los principales verificadores de la calidad del servicio, *se* seguirá un procedimiento establecido para conocer y gestionar esta satisfacción.

Este proceso se integraran todas las áreas y actividades relacionadas a la obtención de información sobre la opinión, dudas y sugerencias de los clientes, esta información será utilizada para conocer la capacidad de respuesta en la resolución de problemas y requisitos.

4.4. Proceso de Comercialización

El proceso de comercialización inicia desde la propuesta del servicio, la cual se la hará de manera escrita, ya que es un servicio completo.

Con el fin de prestar un servicio de calidad y personalizado, se analizará las características de los clientes (objetivos, imagen, posicionamiento, clientes, etc.) con el objetivo de lograr de una integración de los aspectos tecnológicos dentro del funcionamiento y de la imagen de la empresa.

El elemento principal del servicio será la comunicación permanente con el cliente, búsqueda de soluciones en el menor tiempo posible, buscando siempre la optimización de recursos, procurando que el cliente se sienta parte del proceso de prestación del servicio.

Contenido de la prestación del Servicio Test de Penetración

Servicio disponible las 24h a través del portal web, cada usuario tendrá su respectiva cuenta en el sistema.
Chequeo de vulnerabilidades en los sitios web de la empresa, en las cuales se aplicara scanner de vulnerabilidades, fuzzing en las aplicaciones web, rendimiento.
Informe detallado sobre los resultados encontrados, disponible a través de la cuenta otorgada.

4.4.1. Presentación de la Empresa

La imagen de la empresa, desde el logo, el nombre, la ambientación y la decoración estarán acorde con los valores de la empresa y del sector.

En las salidas y visitas a clientes, tanto comerciales cómo técnicas, se cuidará la imagen y la profesionalidad.

La denominación y logo de la empresa se usará en toda la papelería cómo imagen corporativa. Se presentará en el brochure de la empresa.

El logo del servicio será atractivo y fácil de recordar. Se buscara la adaptación con los clientes de la empresa, con servicio de soporte online las 24 horas, todo esto estará automatizado en el portal web.

4.5. Penetración en el mercado

4.5.1. Canales de Distribución Red Comercial

Nos basaremos en la distribución del servicio en forma personal, se contactaran posibles clientes, con los cuales se distribuirá la información

detallada del servicio, es fundamental conocer las necesidades de los clientes, para ofrecer una solución efectiva y eficaz.

En muchas ocasiones la presentación personal permite dar a conocer servicios y posibilidades que los propios clientes ignoran.

Es importante que en la presentación de los servicios se acceda tanto a la persona con autoridad en la decisión final como a los trabajadores de la empresa vinculados a las nuevas tecnologías, que facilitan el proceso de toma de decisión.

Siguiendo esta referencia, para introducirnos en el mercado seguiremos las siguientes estrategias:

- Mailing: Previa selección de los clientes objetivo
- Visitas personalizadas: Previo contacto telefónico. En estas presentaciones se mostrarán trabajos desarrollados por la empresa. El éxito comercial previsto está entre el 10% y 15%.
- Acuerdos de colaboración con agencias de publicidad y marketing
- Acuerdos con asociaciones empresariales para ofrecer nuestros servicios a sus asociados en condiciones ventajosas

La forma de cobro será mediante una suscripción anual en la cual el cliente tendrá un respectivo usuario con el cual podrá acceder al servicio. Aunque también se podría tener membrecías adicionales si el cliente así lo estima.

4.5.2. Acciones de promoción

Las acciones de promoción tendrán los siguientes objetivos.

- Presentar las ventajas de los servicios de la manera más sencilla y atractiva posible.
- Distinguir y poner en valor las diferencias respeto a los competidores y alternativas no especializadas.
- Crear prestigio e imagen de calidad.

Para apoyar estas acciones realizaremos:

- Mailing destinado a las empresas que cumplan el perfil de cliente potencial.
- Presentación personal de los productos mediante CD y DVD.
- Presentación on-line de nuestros servicios mediante invitaciones a conocer nuestra Web.
- Catálogos publicitarios (brochures) en los que se recogerán el producto y servicio que ofrecemos.
- Presencia en directorios comerciales
- Campaña en Internet
- Campaña de publicidad en prensa

El coste aproximado de estas acciones será de ($1442,22) el primer año, inversión que se mantendrá en sucesivos ejercicios.

4.5.3. Prescriptores

Los más importantes prescriptores de estos servicios son los propios clientes que usaran el servicio, al crearnos una buena reputación, ya que este tipo de servicios está enfocado a empresas medianas y grandes, conocida como publicidad de boca en boca.

Por tanto la satisfacción final es imprescindible para que nos recomienden, borrando la imagen de que las nuevas tecnologías no son necesarias, sino enfocándonos en los beneficios a corto y mediano plazo.

Tendremos relación directa con las empresas-cliente específicamente con el personal del área informática, ya que estos influyen en la toma de decisiones a la hora de incorporar nuevas tecnologías en las empresas.

4.6. Recursos Humanos

4.6.1. Estructura de la Organización

La estructura de la organización es simple, debido a que somos pocos trabajadores, presentados en este cuadro:

Promotor – Publicista	1
Programadores	2

Tabla 4.2 Estructura de Organización

Las actividades de cada uno serán detalladas a continuación:
Promotor – Publicista: Responsable de la coordinación de los aspectos comerciales y financieros, contacto al exterior con los clientes.

Programadores: Responsables de mantenimiento y resolución de problemas, encargados del optimo funcionamiento del servicio, del análisis de vulnerabilidades.

4.7. Análisis Financiero

Se detalla a continuación una tabla con los valores de los activos y pasivos de la empresa:

Gastos de constitución y establecimiento	**500**
Aplicaciones Informáticas	**300 (99)**
Equipos de Computación	**2000 (660)**
Mobiliario	**200 (20)**
Hosting	**80**
Internet	**40**

Tabla 4.3 Análisis Financiero

4.7.1. Financiamiento del Proyecto

Los desarrolladores del servicio se encargaran de los gastos al 100%, con sus propios recursos con los cuales se financiaría el proyecto.

4.7.2. Depreciaciones

Se aplicara la depreciación máxima (5 años) para los equipos de computación y el mobiliario, aplicando el método lineal (cuotas fijas), los porcentajes de depreciación se basaran en los brindados por los organismos gubernamentales.

Activos Fijos	% ANUAL
Inmuebles (excepto terrenos), naves, aeronaves, barcazas y similares	5%
Instalaciones, maquinarias, equipos y muebles	10%
Vehículos, equipos de transporte y equipo caminero	20%

móvil	
Equipos de computo y software	33%

Tabla 4.4 Activos Fijos

4.7.3. Ingresos – Previsión de Ventas Anuales

El servicio tendrá un costo de $1200 anuales, este dependerá será sujeto a cambios según las necesidades de la empresa – cliente.

4.7.4. Costos Variables

Dado que es una empresa de servicios de tipo binomio servicios no tenemos costos variables debido a que no incurrimos en el gasto de producción (materia prima).

4.7.5. Costos Fijos

Se detallan continuación los costos fijos:

Hosting	**80**
Internet	**40**
Servicios Básicos (Agua, Luz, Telefono)	**100**
Depreciaciones (por año)	**344 (28.66 mensual)**
Total Costo Fijo:	**248.66**

Tabla 4.5 Costos Fijos

Depreciaciones

Tablas de Depreciación

Mobiliario

Periodo	Depreciación Periodo	Depreciación Acumulada	Valor Libros
0	0	0	200
1	36	36	164
2	36	72	128
3	36	108	92
4	36	144	56
5	36	180	20

Tabla 4.6 Depreciación Mobiliaria

Equipo de Cómputo y Software

Periodo	Depreciación Periodo	Depreciación Acumulada	Valor Libros
0	0	0	2300
1	308,2	308,2	1991,8
2	308,2	616,4	1683,6
3	308,2	924,6	1375,4
4	308,2	1232,8	1067,2
5	308,2	1541	759

Tabla 4.7 Depreciación de Equipos de Computación

4.7.6. Punto de Equilibrio

Margen de contribución= Precio de venta unitario – Costo variable unitario

Margen de contribución: 1200 anuales

Margen de contribución mensual: 100

$$\text{Punto de equilibrio} = \frac{\text{Costo Fijo Total}}{\text{Margen de Contribución}}$$

Punto de equilibrio = 2.48

En conclusión se deberá proporcionar el servicio al menos a 3 empresas – clientes para no tener perdida.

4.7.7. Flujo de Caja

Detalle

Tasa Nominal (8%)	0,8

Rubros (Años)	0	1	2	3	4	5
Ingreso		3600	3600	3600	3600	3600
Gastos de constitucion y establecimiento	500	0	0	0	0	0
Equipos de Computacion	2000	0	0	0	0	0
Servicios Básicos		100	100	100	100	100
Internet – Hosting		120	120	120	120	120
Sueldos		800	800	800	800	800
Depreciaciones		344	344	344	344	344
Subtotal	2500	1364	1364	1364	1364	1364
Inversion	-2500					
Valor de Salvamento						0
Flujo Caja	-2500	2236	2236	2236	2236	2236
FCp/((1+TasaNominal)^p)		1242,22	690,12	383,40	213,00	118,33

Tabla 4.8 Flujo de Caja

4.7.8 Rentabilidad

Al realizar el análisis del punto anterior tenemos los siguientes resultados sobre el Valor Actual Neto (VAN) y la Tasa Interna de Retorno (TIR):

VAN	147,08
TIR	85%

Tabla 4.9 Rentabilidad

Podemos concluir que el valor actual neto es mayor a 0 lo cual significa que el proyecto es rentable y puede aceptarse.

Sobre la tasa interna de retorno tenemos que el valor es mayor a la tasa nominal propuesta (8%) por lo tanto el proyecto es rentable.

5. Implementación del Servicio

Las herramientas que se han sido utilizadas para este proyecto son:

- WebScarab, proyecto OpenSource, alojado en http://www.owasp.org/development/webscarab.
- Jmeter, proyecto de Apache, alojado en http://jakarta.apache.org/jmeter
- OpenVas http://www.openvas.org/

El objetivo de esta herramienta es que pueda utilizarse de forma automática o interactiva para evaluar la seguridad de aplicaciones web.

Nuestro producto llamado, FWEB - TOOL está diseñado para pruebas de estrés y de vulnerabilidades en aplicaciones web, las cuales permiten examinar las cabeceras en las peticiones enviadas y recibidas, se lo puede operar como un proxy de intercepción, que permite al operador revisar y modificar las peticiones creadas por el navegador antes de que sean enviados al servidor, y para revisar y modificar respuestas enviadas por el servidor antes de que sean recibidas por el navegador de tal manera que simula todas las funcionalidades de un Navegador ("Browser"), siendo capaz de manipular resultados en determinada requisición y reutilizarlos para ser empleados en una nueva secuencia.
Además, posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de requisiciones que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción.

FWeb - Tool proporciona un número de plugins los cuales son destinados para generar peticiones (basado en la interacción del usuario, y basados en la información obtenida mediante el análisis de páginas recuperadas por otros plugins).

Entre las características del FWeb – Tool tenemos:

Funcionalidades que provee WebScarab

El plugin de Manual Request se encarga de realizar un análisis pasivo buscando datos controlados por el usuario en los encabezados y cuerpo de las respuestas HTTP para identificar posibles.

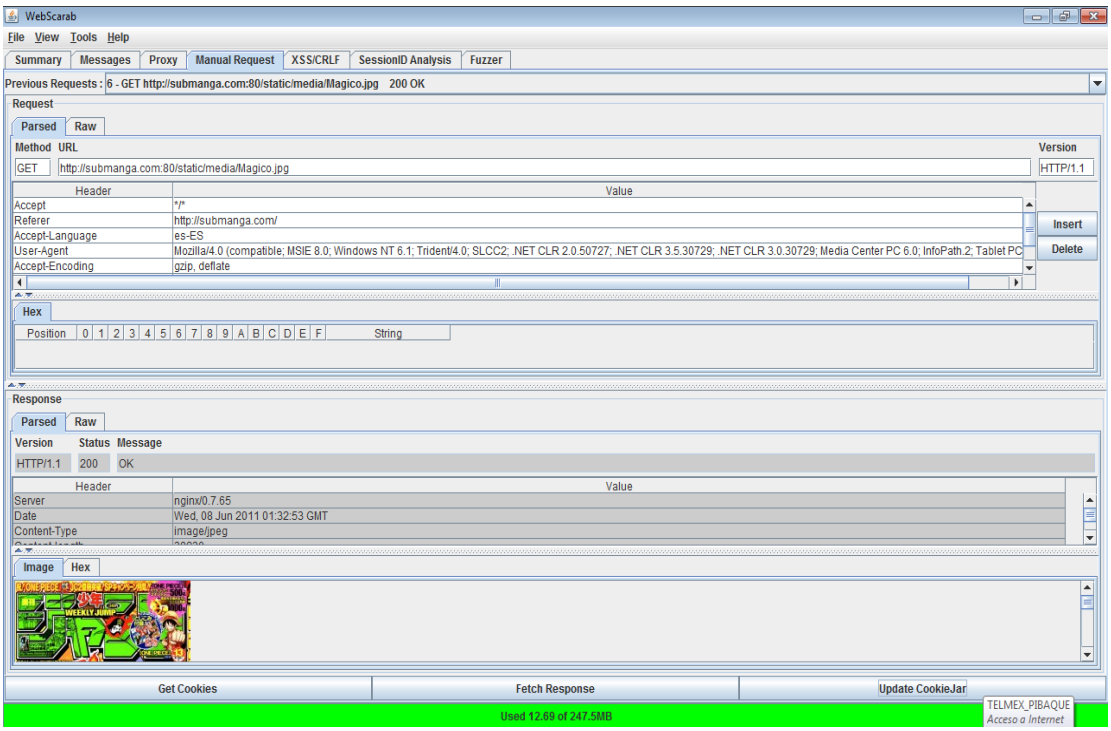


Figura 5.1 Plugin Manual Request

El plugin de XSS/CRLF se encarga de Inyecciones CRLF (partición de respuesta HTTP) y vulnerabilidades de secuencia de comandos en sitios cruzados (XSS).

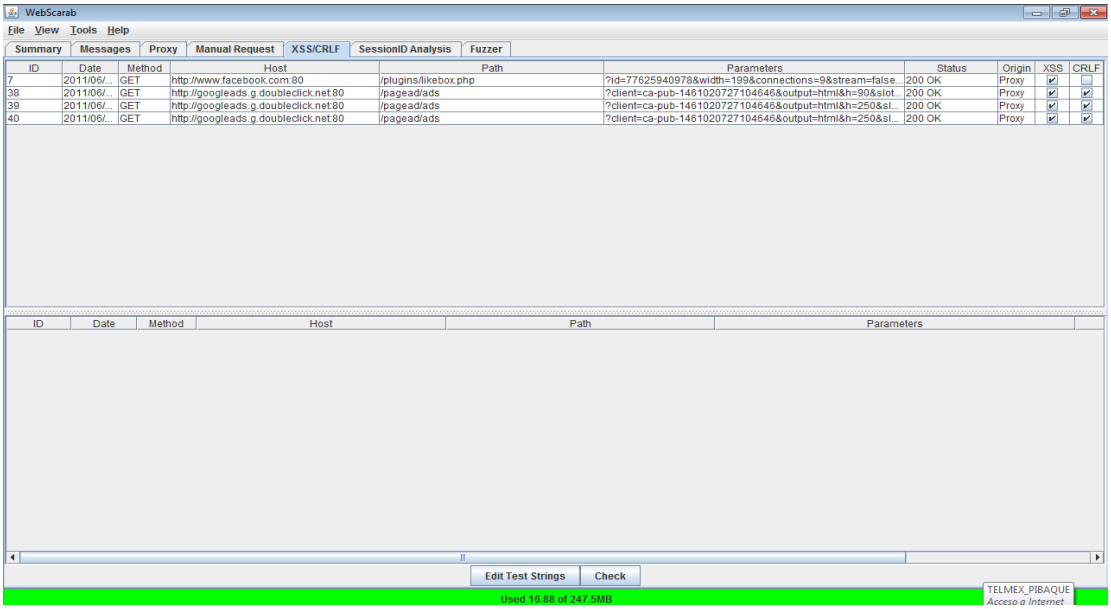


Figura 5.2 Plugin XSS/CRLF

El plugin de SessionID Analysis que recolecta y analiza un número de cookies para poder determinar mediante un grafico el grado de aleatoriedad y predictibilidad.

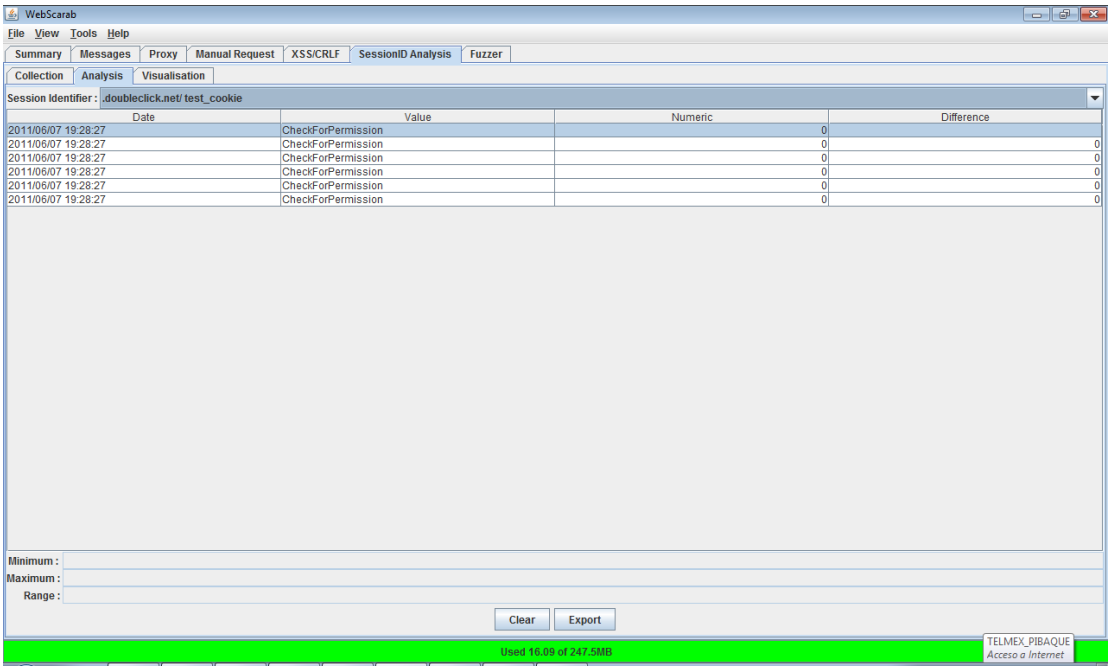


Figura 5.3 Plugin SessionIDAnalysis

El plugin de Fuzzer nos permite un análisis con parámetros definidos previamente, ya sea desde un archivo plano o una variable ingresada.

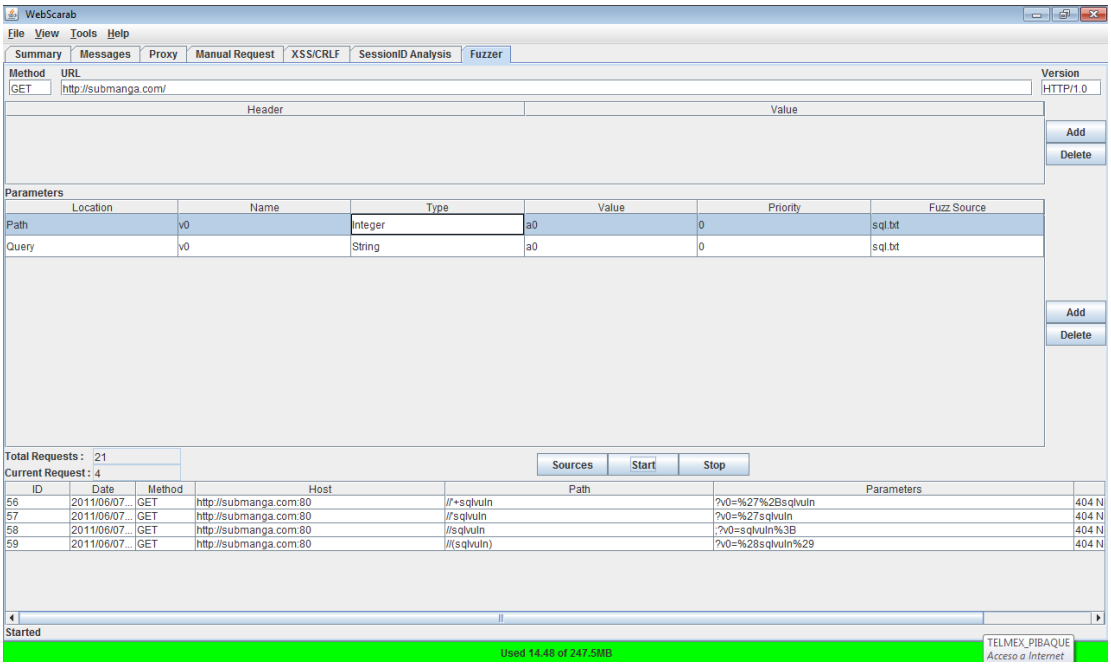


Figura 5.4 Plugin Fuzzer

Funcionalidades que provee Jmeter

Los plugins de PeticionHTTP y Resultados en Árbol permiten al usuario emular una red más lenta, simulando un gran tráfico de usuarios que acceden a dicha pagina y así se puede observar cómo se desempeña el sitio cuando es accedido.

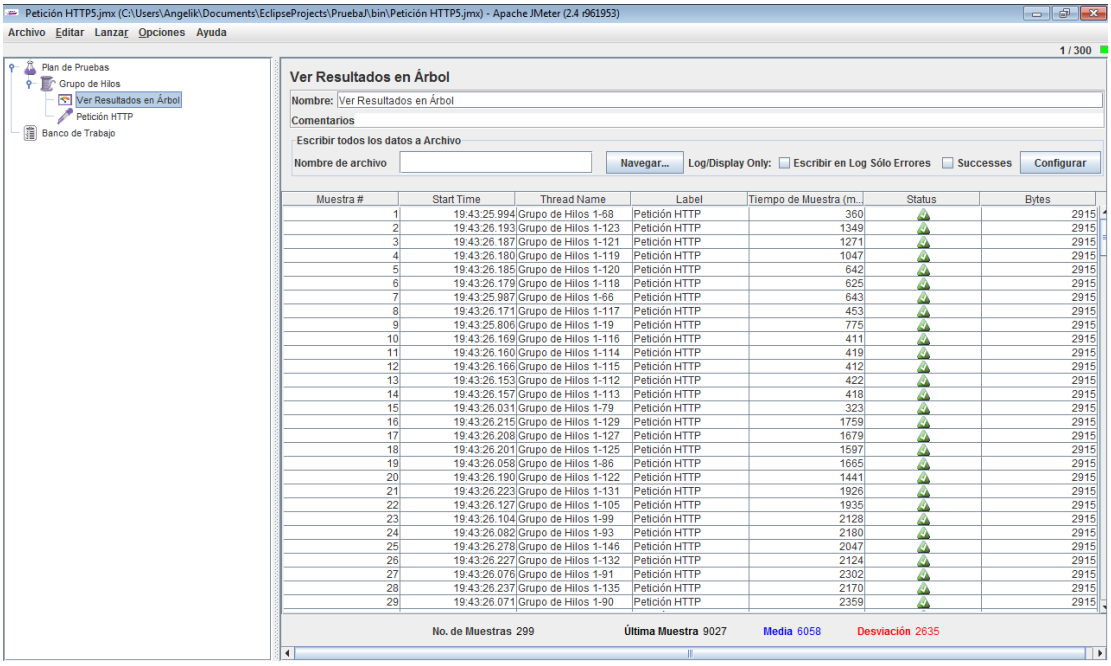


Figura 5.5 Plugins Jmeter

Funcionalidades que provee OpenVAS

OpenVAS es un sistema de análisis de vulnerabilidades y un scanner de seguridad de red el cual nos provee pruebas generales de vulnerabilidades, dándonos un panorama general, ingresando solo la dirección a analizar, tiene una interfaz grafica a modo de cliente, los resultados son analizados en el servidor de OpenVAS.

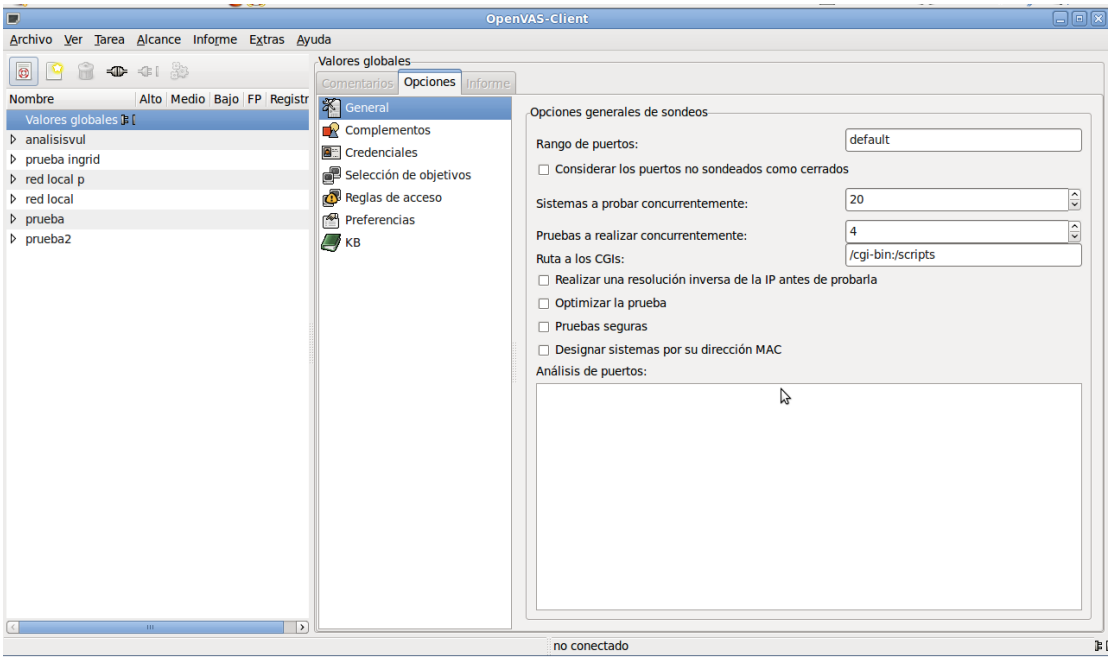


Figura 5.6 OpenVAS

5.1 Pruebas

Mediante las pruebas se intentará reunir la mayor cantidad de información posible sobre el sitio web seleccionado.

Para dar fe del buen rendimiento del servicio web, éste ha sido sometido a pruebas de stress. Se van a realizar peticiones HTTP emulando a usuarios realizando búsquedas de productos.

Su funcionamiento es muy sencillo. Primero hay que indicarle los siguientes parámetros:

- Número de hilos: Equivale al número de usuarios que se desean emular.
- Periodo de subida: Es el lapso de tiempo en segundos que se desea tener entre cada grupo de usuarios.

- Contador del bucle: Utilizado para indicar el número de veces que se va a llevar a cabo la emulación.

Y una vez definidas los parámetros anteriores, hay que indicarle la dirección del servidor al cual van a ir dirigidas las peticiones HTTP.

Para un mejor resultado ha decidido realizar varias pruebas de stress en donde la variable que diferencia es el tiempo entre ciclos que los usuarios tienen para realizar su petición. Así se podrán llegar a saber los límites que puede tener el servidor.

Se le agrega peticiones HTTP para poder ver lo que se recibe y se envía como información, es conveniente agregar Manejadores de Cabeceras para poder grabar valores de las sesiones en caso de que las haya.

Con ayuda de los diferentes tipos de gráficos, se puede hacer un mejor conteo de las muestras sacando un promedio, porcentaje de error, máximo y mínimo. Para así poder hacer un análisis más detallado al momento de mostrar el reporte al usuario.

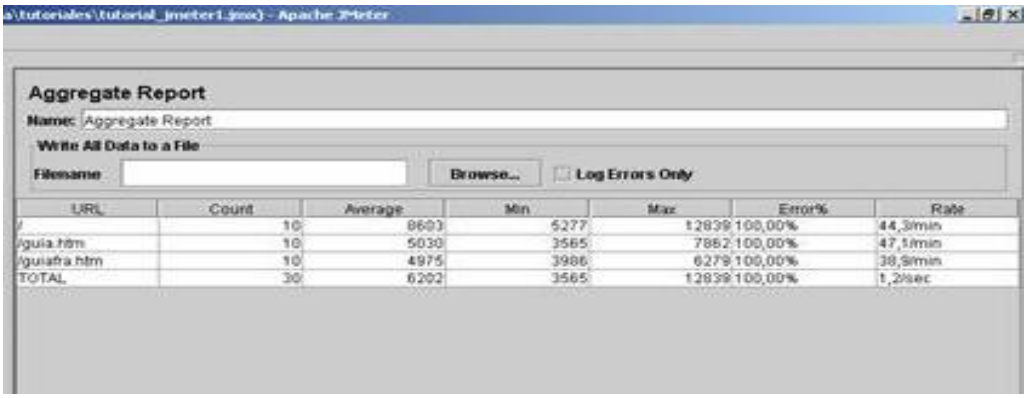


Figura 5.7 Ejemplo Peticiones

Las pruebas se efectuaron usando sitios web vulnerables ante los diferentes tipos de ataques y se almacenan los resultados en la base de datos para poder evaluar los resultados y poder analizarlos.

Nos ayudamos de scripts para simular los diferentes tipos de ataques XSS, de acuerdo a los resultados se puede detectar si el ataque fue o no exitoso. Para cada prueba efectuada se usaban plantillas previamente probadas, para así poder sacar una estadística al momento de analizar los resultados.

5.2 Análisis

El análisis se lo efectúa mediante los resultados de las muestras, se tabula la información y se obtiene un grafico de muestras vs tiempo, el promedio y la desviación estándar. Mediante el uso de todos los escáneres de vulnerabilidades descubiertas se determina si hay cualquier otra vulnerabilidad que podría ser explotada para obtener acceso a un host de destino en una red.

5.3 Modelo de Reporte

- Introducción
    - Fecha llevado a cabo
    - Red de Datos
        - Número de servidores y estaciones de trabajo
        - Detalles del sistema operativo
        - Principales aplicaciones de software
        - Configuración de hardware y la configuración
    - Ámbito de aplicación de la prueba
        - Limitaciones y restricciones impuestas por el equipo
        - Propósito de la prueba
- Detalle de los ataques
    - Problemas de seguridad descubiertos con el nivel de criticidad adecuada especificada

- Posibles Causas
- Cuestiones de seguridad general descubiertas con un nivel de criticidad adecuada especificada

- Vulnerabilidad Definiciones
  - Crítico

    Una vulnerabilidad permite la ejecución remota de código, la elevación de privilegios o una denegación de servicio en un sistema afectado.
  - Importante

    Un fallo de seguridad, cuya explotación puede resultar en el compromiso de la confidencialidad, integridad o disponibilidad de los datos de la empresa.
  - Información de fugas

    Servicios inseguros y protocolos están siendo empleados por el sistema que permita que potencialmente permite un acceso ilimitado a la información sensible.
- Detalles de las herramientas utilizadas.
- Metodología utilizada.
  - Reconocimiento

    El probador intentará reunir la mayor cantidad de información posible sobre la red seleccionada. El reconocimiento puede adoptar dos formas es decir, activos y pasivos. Un ataque pasivo es siempre el mejor punto de partida ya que normalmente iría en contra de los sistemas de detección de intrusos y otras formas de protección, etc.
  - Exploración

    Mediante el uso de todos los escáneres de vulnerabilidades descubiertas se determina si hay cualquier otra vulnerabilidad que podría ser explotada para obtener acceso a un host de destino en una red.

CONCLUSIONES

1. El interés por diferentes tipos de ataques se ha incrementando con rapidez. Muchas empresas están buscando aumentar la eficiencia de sus operaciones y reducir las vulnerabilidades. La oferta de este tipo de soluciones cada vez es mayor, el fuzzing, es una alternativa muy eficaz a la hora de encontrar vulnerabilidades en aplicaciones, sobre todo nuevas vulnerabilidades, pero es preciso tener en cuenta que no existe ninguna herramienta que pueda garantizar la seguridad de un software al 100 %, por lo que el uso de este tipo de soluciones debe ser una parte más de la auditoría de un sistema.

2. El poder del FWEB – TOOL radica en la capacidad de leer peticiones, las cuales pueden ser clasificadas para un control más exhaustivo, identificar las vulnerabilidades y simular una red traficada por muchos usuarios y así calificar el rendimiento de los sitios analizados.

3. Se ha llevado a cabo una aplicación capaz de almacenar en una base de datos la información enviada. Se han realizado unas pruebas de stress sobre el servidor, las cuales han dado unos resultados satisfactorios.

4. El servicio basado en la herramienta descrita es rentable, se podría abrir un nuevo mercado en el futuro y satisfacer la demanda con este servicio.

RECOMENDACIONES

1. Debido a las consecuencias que puede ocasionar el tener un sitio web vulnerable, se recomienda tomar medidas de seguridad exigibles a los ficheros y tratamientos de datos informativos. El hecho de que una aplicación deje de responder puede ser el indicio de que existe alguna forma de controlar el flujo de instrucciones. Si es así, el problema es muy grave y es necesario hacer uso de las respectivas herramientas para controlar la situación.

2. Al ejecutar cualquier herramienta, nos debemos de concentrar en remover las vulnerabilidades detectadas más críticas. Inicialmente será imposible atacar todas las disconformidades en forma simultánea (excepto que se empiece de cero siguiendo las mejores prácticas de testeo).

3. Es útil hacer uso de las herramientas fuzzing en fase de desarrollo para probar los diseños/configuraciones antes de ponerlos en producción y así tener en claro los procedimientos a realizar y hacer un buen uso de la información detallada en los reportes que presenten dichas herramientas.

ANEXOS

ANEXO A

OWASP

Es una aplicación de código abierto del proyecto de seguridad. Es también un cuerpo de estándares emergentes, con la publicación de su primera norma en diciembre de 2008, la seguridad de las aplicaciones de OWASP Verificación estándar (ASVS). El objetivo principal del proyecto OWASP ASVS es normalizar la cobertura y el nivel de rigor en el mercado a la hora de realizar la verificación de seguridad de nivel de aplicación. Crea estándares abiertos comercialmente que se adapten a las tecnologías específicas basadas en la Web.
El estándar proporciona una base para probar la aplicación de controles técnicos de seguridad, así como controles técnicos de seguridad en el ambiente, que se invocan para proteger contra las vulnerabilidades como Croos-Site-Scripting (XSS) y la inyección de SQL.

Los proyectos OWASP se dividen en dos categorías principales: proyectos de desarrollo y proyectos de documentación.

Los **proyectos de documentación** actuales son:

- Guía OWASP – Un enorme documento que proporciona una guía detallada sobre la seguridad de las aplicaciones web.
- OWASP Top 10 – Documento de alto nivel que se centra sobre las vulnerabilidades más críticas de las aplicaciones web.
- Métricas – Un proyecto para definir métricas aplicables de seguridad de aplicaciones web.

- Legal – Un proyecto para ayudar a los vendedores y compradores de software a negociar adecuadamente los aspectos de seguridad en sus contratos.
- Guía de pruebas – Una guía centrada en la prueba efectiva de la seguridad de aplicaciones web.
- ISO 17799 – Documentos de apoyo para organizaciones que realicen revisiones ISO 17799.
- AppSec FAQ – Preguntas y respuestas frecuentes sobre seguridad de aplicaciones web.

Los proyectos de desarrollo incluyen:

- WebScarab – Un aplicación de chequeo de vulnerabilidades de aplicaciones web incluyendo herramientas proxy.
- Filtros de validación (*Stringer* para J2EE, *filters* para PHP) – Filtros genéricos de seguridad perimetral que los desarrolladores pueden usar en sus propias aplicaciones.
- WebGoat – Una herramienta interactiva de formación y benchmarking para que los usuarios aprendan sobre seguridad de aplicaciones web de forma segura y legal.
- DotNet – Un conjunto de herramientas para hacer seguros los entornos .NET.

ANEXO B

Diagramas de Clases

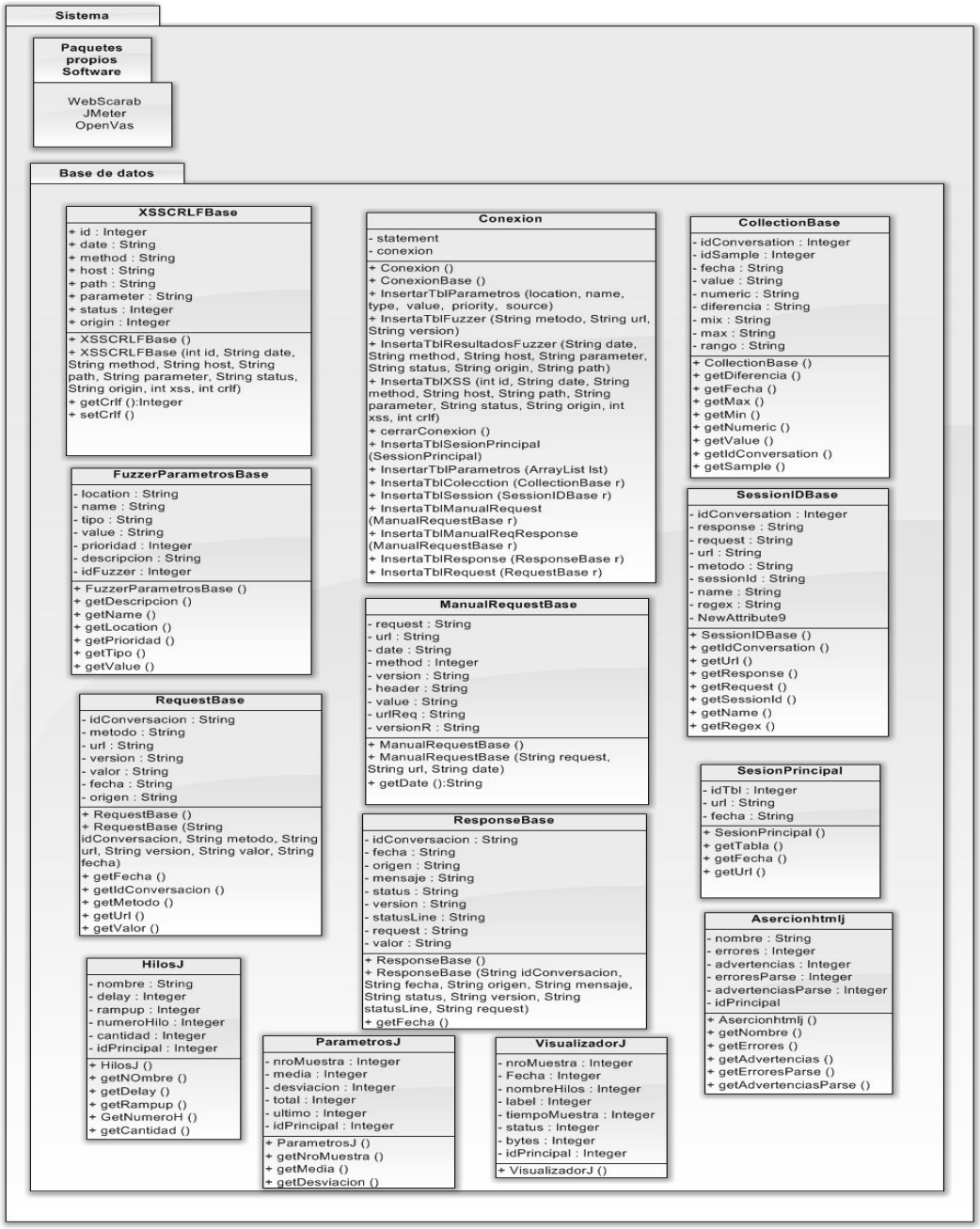


Figura ANEXO B Diagrama de clases

ANEXO C

Informe Detallado

FWEB TOOL

Informe del Análisis de Vulnerabilidades y Stress de aplicaciones Web.

1. **Introducción**

   El presente informe constituye el producto entregable de análisis de seguridad. El mismo que se enfoca en analizar y medir el desempeño de las aplicaciones web usando los protocolos HTTP y HTTPS.

2. **Vulnerabilidades, soluciones y Contramedidas.**

   Las herramientas utilizadas en el scanning de vulnerabilidades fueron las siguientes:

   - WebScarab
   - JMeter
   - OpenVas

3. **Estadísticas Generales**

Número de Usuarios:

Período de Subida (seg.):

Errores de Traspaso:

Graficos

Cookies vs Tiempo

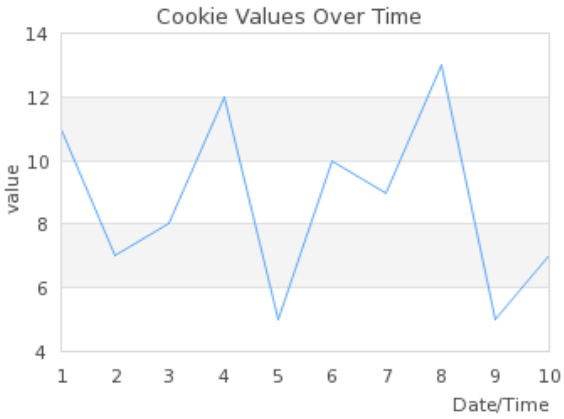


Figura ANEXO C. Cookies vs Tiempo

Stress: Número de Muestras vs Tiempo

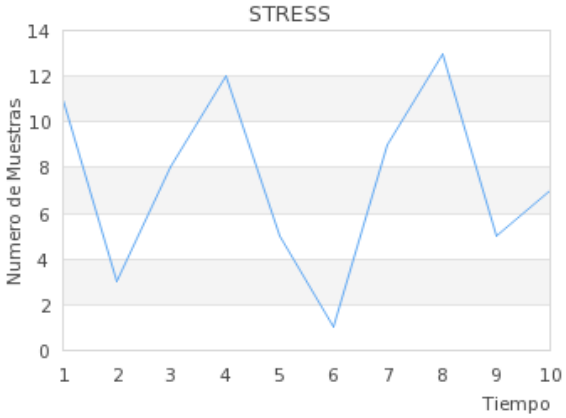


Figura ANEXO C. Nro. Muestras vs Tiempo

ANEXO D

Informe Sitio Web

REPORTE EJECUTIVO

Fecha: Mon May 16 22:35:25 CST 2011

Puerto: 8080

URL: http://www.cidis.espol.edu.ec

IP:192.168.2.5

ANALISIS DE VULNERABILIDADES

FUZZING

Metodo:	GET
Version :	HTTP/1.0
Archivo usado para las vulnerabilidades:	sql.txt
Campo analizado:	Query
Tipo de dato:	String

DETALLE FUZZING

Fecha	Metodo	Url_host	Parametros	Status	Path
Mon Feb 21 11:54:08 COT 2011	GET	www.cidis.espol.edu.ec	v0=%27sqlvuln	200 OK	/administracion.jsp

Mon Feb 21 11:54:08 COT 2011	GET	www.cidis.espol.edu.ec	v0=%28sqlvuln%29	200 OK	/administracion.jsp
Mon Feb 21 11:54:08 COT 2011	GET	www.cidis.espol.edu.ec	v0=sqlvuln%3B	200 OK	/administracion.jsp
Mon Feb 21 11:54:08 COT 2011	GET	www.cidis.espol.edu.ec	v0=%27%2Bsqlvuln	200 OK	/administracion.jsp
Mon Feb 21 11:54:08 COT 2011	GET	www.cidis.espol.edu.ec	v0=a%27+or+1%3D1--	200 OK	/administracion.jsp

MANUAL REQUEST

Total de Request:	33
Total de Response :	34

Total de Request Analizados :	1
Total de Response Analizados :	1

DETALLE REQUEST

Header analizado de la peticion

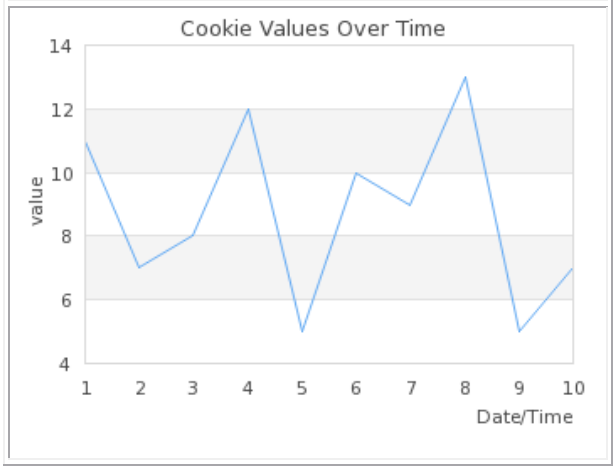
GET http://profile.ak.fbcdn.net:80/hprofile-ak-snc4/211555_100001042864493_3945804_q.jpg HTTP/1.1 Accept: */* Referer: http://www.facebook.com/ Accept-Language: es-ES User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.2) Accept-Encoding: gzip, deflate Host: profile.ak.fbcdn.net Proxy-Connection: Keep-Alive

SESSION ID

Numero de Muestras:	2

Cookie Values

Grafico Cookie



XSS

Valor:	null
Status :	200
Archivo de Origen :	Proxy
Cantidad de Ataques XSS:	no ocurrio ningun ataque XSS
Cantidad de Ataques CRLFInjection:	no ocurrio ningun ataque CRLFInjection

DETALLE XSS

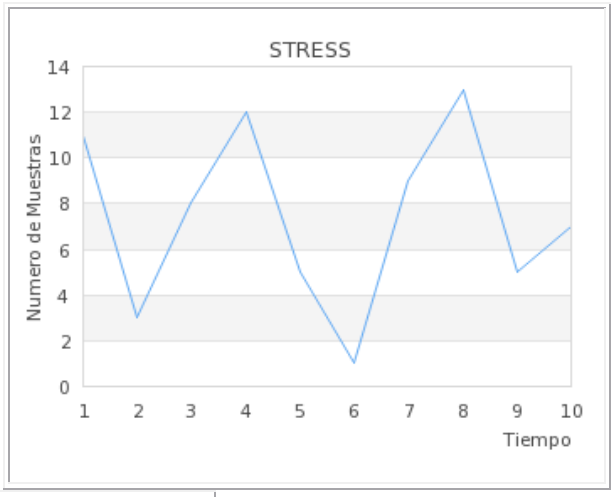
No existen datos para mostrar debido a que no se produjo ningun ataque

VULNERABILIDADES

ANALISIS DE STRESS

Aserciones HTML	
Errores:	0
Advertencias:	0
Advertencias de Traspaso:	4
Errores de Traspaso:	1

Grafico Stress



Resultados

#Muestras	Desviación	Media
100	2058	1152

MEDICION DE STRESS

Cantidad de Usuarios:	2
Atraso:	0
Periodo:	0

Visualizador: Peticiones HTTP

#Muestras	Tiempo(ms)	Status	Tamaño en Bytes
1	1300076803904	OK	2915
1	1300077044947	OK	2915
1	1300424299335	OK	9324

ANEXO E

Diptico

• ¿Qué es Fuzzing?

Se conoce como fuzzing a las diferentes técnicas de pruebas de software capaces de generar y enviar datos secuenciales o aleatorios a una o varias áreas o puntos de una aplicación, con el objeto de detectar defectos o vulnerabilidades existentes en el software auditado.

• ¿Quiénes lo usan?

El fuzzing es usado por compañías de software y proyectos Open Source para mejorar la calidad del software, por investigadores de seguridad para descubrir y publicar vulnerabilidades, por auditores informáticos para analizar sistemas, y, en última instancia, por delincuentes informáticos (hackers) para encontrar agujeros en sistemas y explotarlos de forma secreta.

• ¿Qué es Test de Penetración?

El Test de Penetración, también llamado a veces "hacking ético" es una evaluación activa de las medidas de seguridad de la información. En los entornos de red complejos actuales, la exposición potencial al riesgo es cada vez mayor y hacer a los sistemas seguros se convierte en un auténtico reto.

El Test de Penetración está dirigido a la búsqueda de agujeros de seguridad de forma focalizada en uno o varios recursos críticos, como puede ser el firewall o el servidor Web.

El párrafo con azul puede ser quitado si tu ves que se ve mucho contenido, porque tampoco quiero que se vea con demasiado contenido, quiero que sea claro y conciso para el usuario

Caracteristicas

- Es un proceso automatizado que envuelve manipulación repetida y suplantación de datos a un software objetivo para el proceso.
- Soporta aserciones para asegurarse que los datos recibidos son correctos, por cookies de hilos, configuración de variables y una variedad de reportes.
- Se pueden también revisar las conversaciones (peticiones y respuestas).
- Extrae los scripts y comentarios de las páginas HTML en el momento en que son vistas a través del FWebFuzzer.
- Observa el trafico entre el navegador y el servidor Web

FWEBTOOL

BIBLIOGRAFIA

[1] Michael Sutton, Adam Greene, Pedram Amini, Bruce Force Vulnerability Discovery, Pearson Eduacation, 2008.

[2] Owasp, Guia de Pruebas OWASP Version 3, Fundacion OWASP, 2008

[3] Jose Miguel Esparza Muñoz, Security Research S21sec labs, 2008.

[4] Verizon Business, 15 ataques de seguridad más comunes en las empresas, http://mgluaces.wordpress.com/2009/12/30/los-15-ataques-de-seguridad-mas-comunes-en-empresas/, fecha de consulta agosto 2010.

[5]Owasp, WebScarab, https://www.owasp.org/index.php/Proyecto_WebScarab_OWASP , fecha de consulta agosto 2010.

[6] Osmosis Latina, Jmeter, http://www.osmosislatina.com/jmeter/basico.htm, fecha de consulta enero 2011.

[7] Apache Jakarta Project, Jmeter, http://jakarta.apache.org/jmeter/, fecha de consulta enero 2011.

[8] OpenVas, OpenVas, http://www.openvas.org/, fecha de consulta enero 2011.

[9]SRI, Instructivo Depreciaciones, http://descargas.sri.gov.ec/download/pdf/instructivo_101.pdf, fecha de consulta marzo 2011.