PROPUESTA DE PROCEDIMIENTO PARA LA EJECUCIÓN DE PENTEST DENTRO DEL ESQUEMA DE PRUEBAS DE LAS FÁBRICAS DE SOFTWARE PARA APLICACIONES WEB

MIGUEL CAMILO PÁEZ PIRAZAN

UNIVERSIDAD PILOTO DE COLOMBIA
PROGRAMA DE POSGRADOS
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ D.C.
2014

PROPUESTA DE PROCEDIMIENTO PARA LA EJECUCIÓN DE PENTEST DENTRO DEL ESQUEMA DE PRUEBAS DE LAS FÁBRICAS DE SOFTWARE PARA APLICACIONES WEB

MIGUEL CAMILO PÁEZ PIRAZAN

Proyecto de grado

Asesor:
César Iván Rodríguez

UNIVERSIDAD PILOTO DE COLOMBIA
PROGRAMA DE POSGRADOS
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ D.C.
2014

AGRADECIMIENTOS

Se dan gracias a la fábrica de software Zone software por permitirme realizar y proponer el procedimiento que se desarrollo en este proyecto de grado, de igual manera al gerente Ingeniero Julian Marin, quién sin su apoyo y comprensión no hubiera sido posible el desarrollo de este proyecto.

CONTENIDO

LISTA DE CUADROS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

Pentest: “Es como comúnmente se denomina a los (Test de penetración) y son en conjunto la forma de denominar a una serie de técnicas utilizadas para evaluar la seguridad de redes, sistemas de computación y aplicaciones involucradas en los mismos.” [1]

Hacker: Este nombre fue dado a las personas que tenían grandes conocimientos sobre las herramientas tecnológicas, lenguajes de programación o redes. “Hacker es un término usado por algunos para definir a un (programador inteligente), y por otros significa (alguien que trata de penetrar dentro de los sistemas informáticos)”.[2]

Hacker de sombrero blanco: Dentro de la definición de Hacker, existen los hackers de sombrero blanco, los cuales son personas con altos conocimientos en temas tecnológicos pero que actual con un esquema ético dentro del mundo hacker, de aquí surgen los llamados Hackers éticos, quienes se encargan de verificar las vulnerabilidades dentro de un sistema de información, red, infraestructura tecnológica y la comunica más no toma ventaja de dichas vulnerabilidades para realizar ataques.

Hacker de sombrero negro: Un hacker de sombrero negro es todo lo contrario a un hacker de sombrero blanco, pues el actuar de este tipo de hackers es el de realizar ataques a los sistemas de información a los cuales les encuentran vulnerabilidades para sacar provecho a estas, ya sea con fines personales o de una comunidad hacker.

Sistema de Información: Un sistema de información es un conjunto de elementos que trabajan entre si ya sea para almacenar, transformar o mostrar información que haya sido manipulada por el mismo.

C Sharp (C#): Es un lenguaje de programación moderno, de alto nivel, de múltiples paradigmas y de uso general para crear aplicaciones con Visual Studio y .NET

---

[1] E-Securing C.A. ¿Qué es un Pen-Test? Herramientas de Pen-testing [en línea]. <http://www.e-securing.com/novedad.aspx?id=55> [Citado en (2010]

[2] Search Security. Hacker [en línea]. <http://searchsecurity.techtarget.com/definition/hacker> [Citado en (2006)].

Framework. C# se diseñó para que fuera simple, poderoso, con seguridad de tipos y orientado a objetos.[3]

Kali de Linux: Es un Proyecto open source que es mantenido y fue fundado por Offensive and security, un proveedor de clase mundial de entrenamiento en seguridad informática y servicios de test de penetración[4]. Kali de linux es una distribución GNU/Debian diseñada principalmente para auditoria y seguridad informática en general.

Web Scarab: Webscarab es un marco de trabajo para analizar aplicaciones web que se comunica usando los protocolos HTTP y HTTPS. Está escrito en Java, por lo que es portable a muchas plataformas. WebScarab tiene muchos modos de operación, implementados por varios plugins. Su uso más común es operar WebScarab 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. [5]

Nmap: Es un programa de código abierto, que se usa para escanear puertos, servicios y descubrir servidores en una red informática.[6]

Zenmap: Es el GUI oficial de seguridad de Nmap, se trata de una plataforma multi (Linux, Windos, Mac Os entre otros), el cual tiene como objetivo tener a disposición las mismas funcionalidades de nmap, pero de un modo gráfico y más sencillo de utilizar para principiantes y expertos en nmap.[7]

Openssl: Es una herramienta de criptografía que implementa SSL y TLS protocolos de red y los estándares requeridos por estos. Este software es desarrollado y mantenido por una comunidad abierta de voluntario con gran conocimiento en seguridad de la información.[8]

---

[3] Visual C# Developer Center, Frequently Asked Questions About Visual C# .NET 2003 [En línea] <http://msdn.microsoft.com/es-es/vstudio/hh341490> [Citado en (2005)].

[4] Kali Linux TM. About Kali Linux [en línea]. <http://www.kali.org/about-us/>. [Citado en 2014]

[5] Proyecto Web Scarab [en línea]. <https://www.owasp.org/index.php/Proyecto_WebScarab_OWASP>. [Citado en 2015]

[6] Nmap Org. Guía de Referencia de Nmap [en línea]. < http://nmap.org/man/es/>. [Citado en 2015]

[7] Nmap Org. Nmap Security Scanner [en línea]. < www.nmap.org/zenmap/>. [Citado en 2015]

[8] OpenSSL Criptography and SSl/ TLS tools. About the Openssl Project [en línea]. <https://www.openssl.org/about/>. [Citado en 2015]

Accunetix: Accunetix es una herramienta que puede ser capaz de escanear sitios web en busca de fallos de seguridad que puedan poner en peligro la integridad de un sitio web publicado en internet.[9]

SQLMAP: Sqlmap es una herramienta de pentesting open source que automatiza la el proceso de detección y explotación de fallas de sql injection.[10]

Nikto: Nikto es un web server scanner que realiza test contra aplicaciones web, usando múltiples validaciones varios items, como lo son archivos y directorios potencialmente peligrosos, validación de actualizaciones entre otros.[11]

[9] Accunetix: Web a prueba de hackers. [en línea]. < http://www.acunetix.com/vulnerability-scanner/TCNreview.pdf>. [Citado en 2015]

[10] SQLMAP: Automatic SQL injection and database takeover tool. [en línea]. <http://sqlmap.org/>. [Citado en 2015]

[11] CIRT: Suspicion Breeds Confidence. [en línea]. <https://www.cirt.net/Nikto2>. [Citado en 2015]

RESUMEN

La Ingeniería de sistemas requiere el estudio de diversas técnicas para desarrollar herramientas de gestión y control, basados en la estadística y procesamiento de la información. El presente proyecto de grado se plantea en el ámbito de la seguridad de la información, enfocándose específicamente en los factores relevantes como lo son la disponibilidad, confidencialidad e integridad de la información.

Se busca proponer un procedimiento adecuado, con el cual las fábricas de software se puedan apoyar, dicho procedimiento será enfocado hacia la eficiencia y eficacia, teniendo en cuenta que no será un procedimiento extenso, puesto que la idea del mismo es que pueda ser aplicado dentro del ciclo de vida del desarrollo sin que se vean impactados los tiempos.

INTRODUCCIÓN

El presente documento contiene el desarrollo de un procedimiento de pentesting que es propuesto y contiene puntos clave para tener unos requerimientos mínimos de seguridad en sistemas de información. Dicho procedimiento será creado a partir de la experiencia que se ha obtenido a partir de varios años de trabajo en fábricas de software y apoyándonos sobre métodos ya preestablecidos por estándares tomando de estos los puntos más importantes para tratar de realizar un cubrimiento general sobre las herramientas.

Este procedimiento propuesto que fue creado se realizó teniendo en cuenta el ciclo de vida del desarrollo de software (análisis, diseño, implementación y pruebas), con el fin de poder realizar la implementación del procedimiento en la etapa de pruebas.

Se pretende con este procedimiento propuesto realizar así una identificación temprana de vulnerabilidades sobre los sistemas de información antes de las entradas a producción, por ende el procedimiento cumple con algunas particularidades, es un procedimiento compacto, pues no se pretende realizar el desarrollo completo de una metodología con este, al ser esto muy costoso en dinero y tiempo, por ende la propuesta será de un procedimiento con el cual se pretenden cubrir puntos importantes en la seguridad de las aplicaciones, el cual no sea extenso en ejecución, para no afectar así los tiempos ya establecidos por la fábrica y el cliente, es decir una propuesta eficiente y eficaz.

1. DEFINICIÓN DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

Actualmente la mayoría de las empresas han catalogado a sus sistemas de información y las plataformas como su activo más preciado, porque estos pueden ser generadores de nuevos datos importantes o ser repositorio de los mismos diariamente. Es por esto que los sistemas de información pueden correr el riesgo de ser vulnerados si no cuentan con unos parámetros de seguridad mínimos que pueden ser cubiertos con pruebas de penetración realizados sobre los sistemas de información.

Algunas de las empresas que contratan el desarrollo de un sistema de información con fábricas de software no contratan los servicios de pruebas de penetración, porque puede resultar costoso este proceso, no solo en tiempo sino en dinero. Por ende, los sistemas de información salen de la fábrica de software a producción solamente con las pruebas funcionales realizadas sobre estos por el equipo de pruebas de la empresa. Adicional a esto, como las empresas contratantes no están interesadas en este tipo de pruebas, las fábricas de software, no tiene un procedimiento dentro del esquema de pruebas que contenga pentest para mitigar el riesgo a posibles ataques informáticos en un futuro cuando estos sistemas de información se encuentren desplegados en producción y expuestos en la WEB.

1.2 FORMULACIÓN DEL PROBLEMA

¿Se puede mediante la implementación de un procedimiento para la ejecución de pentest minimizar riesgos de posibles ataques a los sistemas de información Web desarrollados por fabricas de software?

1.3 JUSTIFICACIÓN

Teniendo en cuenta que muchas empresas cuenta con activos importantes y que la información ha sido catalogada como uno de estos, es de vital importancia resguardar y minimizar el riesgo de que existan vulnerabilidades que puedan ser explotadas por delincuentes informáticos que puedan apropiarse de los datos. Por esta razón es de vital importancia que los sistemas de información y la infraestructura que es usada por las empresas estén preparadas para afrontar ataques, evitando por ejemplo que sus sistemas de información puedan ser la entrada por la cual se puedan cometer dichos delitos. Es por esto que es importante

la ejecución y verificación por medio de pruebas de penetración a los sistemas de información previos a la puesta en producción de estos.

1.4 ALCANCE

Este proyecto es realizado con el fin de proponer dentro del esquema de pruebas de las fábricas de software un procedimiento para realizar pentest a los sistemas de información, con lo cual, se busca tener unos requerimientos mínimos de seguridad sobre los desarrollos para evitar posibles ataques.

Con dicho procedimiento propuesto se pretende:

- Validar requerimientos mínimos de seguridad sobre los sistemas de información.
- Disminuir superficie de ataque a las aplicaciones.

Con el procedimiento que será propuesto se pretende crear una solución ágil y eficiente que pueda ser usada en las fábricas de software sin que estas impacten sutacialmente los tiempos y el dinero que son requerido en el desarrollo de los sistemas de información, por ende el procedimiento abarcará a groso modo los puntos más importantes a tener en cuenta, disminuyendo de este modo la superficie de ataque que se puede dar sobre los sistemas de información en ambientes productivos. Esto con el fin de sintetizar y evitar realizar un procedimiento completo de seguridad sobre las aplicaciones, teniendo en cuenta que esto es un gran costo de tiempo y dinero.

1.5 OBJETIVOS

1.4.1 General

Proponer un procedimiento para la ejecución de pentest dentro del esquema de pruebas de las fábricas de software.

1.4.2 Específicos

- Planear y diseñar un procedimiento para la ejecución de las pruebas de penetración.
- Probar el procedimiento propuesto sobre un sistema de información bajo un ambiente de pruebas controlado.

- Presentar resultados sobre vulnerabilidades encontradas en el sistema de información al cual se le realizaron las pruebas con el procedimiento de pentest propuesto.

2. MARCO REFERENCIAL

2.1 MARCO TEÓRICO

Este proyecto será basado en la creación de una propuesta de procedimientos para realizar pruebas de penetración sobre sistemas de información web, con el fin de minimizar los riesgos que pueda tener un sistema de información en etapas tempranas, como lo es la etapa de ejecución de pruebas de las fábricas de software, pues actualmente algunas fábricas de software no tiene dentro del esquema de pruebas contemplados procedimientos para la ejecución de pentest en sus sistemas de información, por ende si algún sistema de información llegara a tener vulnerabilidades de seguridad serían con estas instalados en los ambientes de producción.

La propuesta de procedimiento que será creada se realizará no solo para minimizar los riesgos que se pueden presentar en los sistemas de información sino también para generar conciencia y unas mejores prácticas en los desarrollos que son generados por las fábricas de software a partir de las vulnerabilidades encontradas.

Actualmente las fábricas de software, cuentan con ambientes de pruebas para las aplicaciones web dentro de los servidores de las empresas en los cuales se están generando constantemente los despliegues de los sistemas de información que están siendo desarrollados y sobre los cuales se generan las pruebas funcionales. Por ende y para efectos de este proyecto se realizarán dichas pruebas sobre un ambiente simulado sobre el cual se ejecutarán las pruebas de penetración al sistema de información.

Para la seguridad en las aplicaciones se deben tener en cuenta varios aspectos, tales como lo son:

- Seguridad en el cliente.
- Seguridad en el servidor.
- Seguridad en la aplicación.
- Seguridad en la comunicación.

Según el OWASP (Open Web Application Security Project), en inglés (Proyecto abierto de seguridad de aplicaciones web), existe un listado de 10 tipos de vulnerabilidades más comunes que aún se pueden explotar en el año 2013.

2.1.1 Inyección (1 de 10): "Los errores de inyección, tales como SQL, OS (comandos de Sistema operativo) y LDAP (Directorio de acceso ligero a un directorio de servicios basado en TCP/IP) ocurren cuando datos no verificados son enviados a un intérprete como parte de un comando o un query"[12].

2.1.2 Administración de sesión y autenticación (2 de 10): Este tipo de defecto se da cuando la aplicación desarrollada no maneja correctamente las sesiones o la autenticación, esto puede ser cuando por ejemplo se están pasando estos parámetros por medio de la URL o cuando por ejemplo nos vamos autenticar no existe algún tipo de cifrado cuando se envían los datos por un medio de comunicación.[13]

2.1.3 Cross-site scripting (XSS) (3 de 10): Este defecto ocurre cuando un atacante envía por medio de la URL data que no ha sido previamente validada, dicha data que se puede enviar puede ser código javascript, al permitir este tipo de inyecciones de código malicioso el atacante puede afectar a la víctima cuando ejecuta dicho código en la máquina de él, puede haber robo de datos, cookies o re direccionamiento a sitio maliciosos. [14]

2.1.4 Objetos referenciados directamente (4 de 10): Este tipo de defecto se presenta cuando en la aplicación que se desarrolló se están exponiendo directamente los objetos, por ejemplo cuando una variable que será ejecutada directamente en una consulta SQL se expone directamente en la URL, el atacante puede cambiar dicho valor por cualquier otro para tratar de identificar a ensayo y error información vital (Ver ejemplo en Figura 1).[15]

[12] OWASP. 2013 Top 10 List [en línea]. <https://www.owasp.org/index.php/Top_10_2013-Top_10> [Citado en 2013]
[13] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10
[14] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10
[15] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10

Figura 1 Ejemplo de Objetos referenciados directamente

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt = connection.prepareStatement(query , … );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

```
http://example.com/app/accountInfo?acct=notmyacct
```

Fuente.https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

2.1.5 Configuración incorrecta (5 de 10): Este tipo de error corresponde a los inconvenientes que se presentan cuando no existe una configuración bien definida y desarrollada para la aplicación, tales como framework, servidor de aplicaciones, servidor web, servidor de base de datos, y plataformas, adicional a esto la falta de actualizaciones en los servidores también puede influir.[16]

2.1.6 Información sensitiva expuesta (6 de 10): Este error corresponde a la falta de cifrado de los datos sensibles, como claves, id de usuario, así como el mal uso de variables, exponiendo por ejemplo en las URL datos importantes para inicios de sesión.[17]

2.1.7 No validación de niveles de seguridad (7 de 10): Algunas aplicaciones web, validan el rol y niveles de seguridad de una usuario antes de mostrar la interface de usuario, sin embargo, esta validación debe ser realizada también en cada momento que el usuario acceda a alguna de las funciones del sistema, no solo realizar una validación inicial, por ende los atacantes pueden forzar la entrada a otros módulos sin la debida autorización.[18]

[16] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10
[17] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10
[18] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10

2.1.8 Cross-Site request forgery (CSRF) (8 de 10): Este tipo de ataque se da cuando un atacante envía un usuario a un sitio web falso, mediante el cual puede realizar el robo de información importante, tal como lo son cookies de inicio de sesión, o dentro del mismo sitio realizar peticiones al usuario para que revele su usuario y contraseña.[19]

2.1.9 Usar componentes con vulnerabilidades conocidas (9 de 10): Usar componentes de los cuales se sabe que tienen vulnerabilidades, como lo pueden ser librerías, frameworks entre otros puede generar errores, pues los atacantes al conocer estas vulnerabilidades pueden llevar a cabo un ataque, el cual puede terminar muchas veces en la caída de servidores, revelación de datos confidenciales entre otros inconvenientes.[20]

2.1.10 Re direccionamientos inválidos a otros sitios dentro de un portal (10 de 10): Muchas veces cuando se realizan los re direccionamientos a otros sitios dentro de un sistema de información web, no se valida correctamente el otro sitio generando así posibles riegos a los usuarios, tales como posibles ataques de phishing, enviar a un usuario a un sitio con malwares, lo cual puede llevar a la perdida de información o creación de virus dentro de las máquinas de los usuarios.

2.1.11 Owasp (Open Web Application Security Project): El proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés) es una comunidad abierta dedicada a habilitar a las organizaciones para desarrollar, comprar y mantener aplicaciones confiables. Todas las herramientas, documentos, foros y capítulos de OWASP son gratuitos y abierto a cualquiera interesado en mejorar la seguridad de aplicaciones.[21]

OWASP es una metodología enfocada hacia, desarrolladores de software, analistas de pruebas y especialistas de seguridad en busca de un mismo fin, el cual es mantener y tener aplicaciones seguras, teniendo en cuenta que no existe una seguridad total, ni 0 porciento de riesgo, sin embargo si puede reducirse mediante el uso de controles.

Teniendo en cuenta lo antes descrito se aclara que OWASP divide el test de penetración en dos fases:

---

[19] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10

[20] Ibid., Disponible en internet: https://www.owasp.org/index.php/Top_10_2013-Top_10

[21] Owasp org. Sobre OWASP: Introducción En: OWASP Books [En línea]. Disponible en: <https://www.owasp.org/index.php/Sobre_OWASP > [Citado en: 26 de Agosto de 2013]

Modo Pasivo: En la fase pasiva se realiza mediante algunas herramientas y conocimiento propio la recopilación de información en esta fase se usa solamente un dominio llamado (Recopilación de información).

Modo activo: En la fase activa el tester iniciara a validar la aplicación teniendo en cuenta la metodología y los controles que contiene cada una.

- Test de gestión de la configuración.
- Test de autenticación.
- Test de autorización.
- Test de gestión de sesiones.
- Test de autorización.
- Test de validación de datos de entrada.
- DOS testing.
- Test de web services.
- Test AJAX.

A continuación se listan los controles a tener en cuenta en la validación del aplicativo, en la siguiente lista solamente se mostrarán algunos de los controles a usar, teniendo en cuenta que no serán usados todos los 66 controles descritos por OWASP para efectos de este proyecto (Ver Cuadro 1).

Cuadro 1. Controles de OWASP

Category	Númerro de referencia	Test Name	Vulnerability
Recopilación de información (Information gathering)	OWASP-IG-001	Spiders, Robots and Crawlers	N.A.
	OWASP-IG-002	Search Engine - Discovery/Reconnaissance	N.A.
	OWASP-IG-003	Identify application entry points	N.A.
	OWASP-IG-004	Testing for Web Application fingerprint	N.A.
	OWASP-IG-005	Application Discovery	N.A.
	OWASP-IG-006	Analysis of Error Codes	Information Disclosure

Cuadro 2. (Continuación)

Gestion de configuración (Configuration	OWASP-CM-001	SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert.	SSL Weakness
	OWASP-CM-002	DB Listener Testing	DB Listener weak
	OWASP-CM-003	Infrastructure Configuration Management Testing	Infrastructure Configuration management weakness
	OWASP-CM-004	Application Configuration Management Testing	Application Configuration management weakness
	OWASP-CM-005	Testing for File Extensions Handling	File extensions handling
	OWASP-CM-006	Old, backup and unreferenced files	Old, backup and unreferenced files
	OWASP-CM-007	Infrastructure and Application Admin Interfaces	Access to Admin interfaces
	OWASP-CM-008	Testing for HTTP Methods and XST	HTTP Methods enabled, XST permitted, HTTP Verb
Test de Autenticación (Authentication Testing)	OWASP-AT-001	Credentials transport over an encrypted channel	Credentials transport over an encrypted channel
	OWASP-AT-002	Testing for user enumeration	User enumeration
	OWASP-AT-003	Testing for Guessable (Dictionary) User Account	Guessable user account
	OWASP-AT-004	Brute Force Testing	Credentials Brute forcing
	OWASP-AT-005	Testing for bypassing authentication schema	Bypassing authentication schema
	OWASP-AT-006	Testing for vulnerable remember password and pwd reset	Vulnerable remember password, weak pwd reset
	OWASP-AT-007	Testing for Logout and Browser Cache Management	Logout function not properly implemented, browser cache weakness
	OWASP-AT-008	Testing for CAPTCHA	Weak Captcha implementation
	OWASP-AT-009	Testing Multiple Factors Authentication	Weak Multiple Factors Authentication
	OWASP-AT-010	Testing for Race Conditions	Race Conditions vulnerability

Cuadro 3. (Continuación)

Test de gestión de sesiones (Session managment)	OWASP-SM-001	Testing for Session Management Schema	Bypassing Session Management Schema, Weak Session Token
	OWASP-SM-002	Testing for Cookies attributes	Cookies are set not 'HTTP Only', 'Secure', and no time validity
	OWASP-SM-003	Testing for Session Fixation	Session Fixation
	OWASP-SM-004	Testing for Exposed Session variables	Exposed sensitive session variables
	OWASP-SM-005	Testing for CSRF	CSRF
Test de Autorización (Autorization testing)	OWASP-AZ-001	Testing for Path Traversal	Path Traversal
	OWASP-AZ-002	Testing for bypassing authorization schema	Bypassing authorization schema
	OWASP-AZ-003	Testing for Privilege Escalation	Privilege Escalation
Bussiness Logic testing	OWASP-BL-001	Testing for business logic	Bypassable business logic
Validación de datos de entrada (Data Valdiation Testing)	OWASP-DV-001	Testing for Reflected Cross site scripting	Reflected XSS
	OWASP-DV-002	Testing for Stored Cross Site scripting	Stored XSS
	OWASP-DV-003	Testing for DOM based Cross Site Scripting	DOM XSS
	OWASP-DV-004	Testing for Cross Site Flashing	Cross Site Flashing
	OWASP-DV-005	SQL Injection	SQL Injection
	OWASP-DV-006	LDAP Injection	LDAP Injection
	OWASP-DV-007	ORM Injection	ORM Injection
	OWASP-DV-008	XML Injection	XML Injection
	OWASP-DV-009	SSI Injection	SSI Injection
	OWASP-DV-010	XPath Injection	XPath Injection
	OWASP-DV-011	IMAP/SMTP Injection	IMAP/SMTP Injection
	OWASP-DV-012	Code Injection	Code Injection
	OWASP-DV-013	OS Commanding	OS Commanding
	OWASP-DV-014	Buffer overflow	Buffer overflow
	OWASP-DV-015	Incubated vulnerability Testing	Incubated vulnerability Testing
	OWASP-DV-016	Testing for HTTP Splitting/Smuggling	Testing for HTTP Splitting/Smuggling

Cuadro 4. (Continuación)

DOS Testing (Denial of Service	OWASP-DS-001	Testing for SQL Wildcard attacks	SQL Wildcard vulnerability
	OWASP-DS-002	Locking Customer Accounts	Locking Customer Accounts
	OWASP-DS-003	Testing for DoS Buffer overflows	Buffer Overflows
	OWASP-DS-004	User Specified Object Allocation	User Specified Object Allocation
	OWASP-DS-005	User Input as a Loop counter	User Input as a Loop Counter
	OWASP-DS-006	Writing User Provided Data to Disk	Writing User Provided Data to Disk
	OWASP-DS-007	Failure to Release Resources	Failure to Release Resources
	OWASP-DS-008	Storing too Much Data in Session	Storing too Much Data in Session
Test de Web Services (Web Services Testing)	OWASP-WS-001	WS Information Gathering	N.A.
	OWASP-WS-002	Testing WSDL	WSDL Weakness
	OWASP-WS-003	XML Structural Testing	Weak XML Structure
	OWASP-WS-004	XML content-level Testing	XML content-level
	OWASP-WS-005	HTTP GET parameters/REST Testing	WS HTTP GET parameters/REST
	OWASP-WS-006	Naughty SOAP attachments	WS Naughty SOAP attachments
	OWASP-WS-007	Replay Testing	WS Replay Testing
Test a Ajax (Ajax Testing)	OWASP-AJ-001	AJAX Vulnerabilities	N.A.
	OWASP-AJ-002	AJAX Testing	AJAX weakness

Fuente. https://www.owasp.org/index.php/The_OWASP_Testing_Framework

Teniendo en cuenta que la metodología OWASP está dirigida hacia aplicaciones, se debe tener presente que dicha metodología plasma en su guía un framework (Ver Figura 2) a seguir, el cual está distribuido en 5 partes:

Antes de iniciar el desarrollo

- Revisar políticas y estándares.
- Generar criterios de medida y métricas (trazabilidad).

Durante la definición y diseño del sistema de información

- Evaluar requerimientos de seguridad.

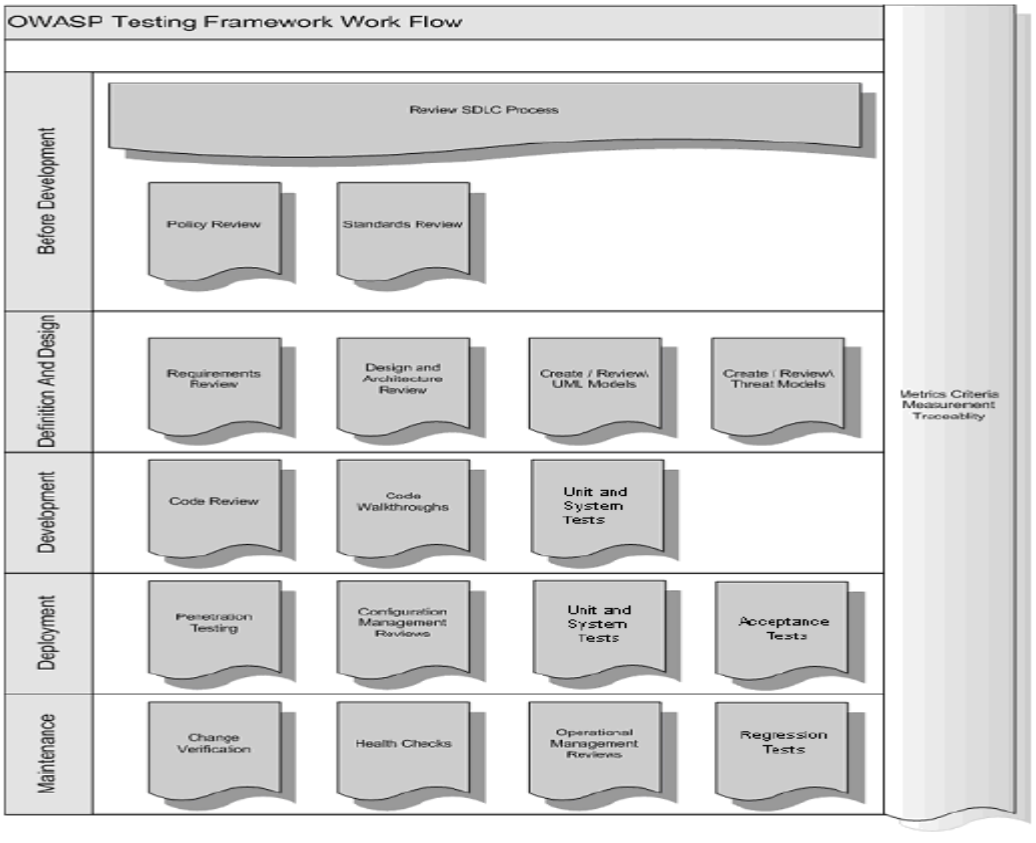
- Evaluar diseño y arquitectura.
- Diseñar y evaluar modelo a usar.
- Crear y evaluar modelo de amenazas.

Durante el desarrollo
- Revisión de la estructura del código.
- Revisión del código.

Durante el despliegue
- Pruebas de penetración a la aplicación.

Durante el mantenimiento y la operación
- Evaluar cómo se administra la parte operacional (Infraestructura vs aplicación).
- Realizar controles periódicos.
- Verificar cambios sobre el aplicativo.

Figura 2 OWASP Framework



Fuente. https://www.owasp.org/index.php/The_OWASP_Testing_Framework

2.1.12 Pruebas de penetración: Las pruebas de penetración son test que se realizan con el objetivo de ver en qué estado se encuentran los sistemas de TI en búsqueda de vulnerabilidades que puedan ser explotadas por un atacante, mediante ataques controlados, ya sea a los sistemas de información de la empresa o a la infraestructura. Para efectos de este proyecto, las pruebas de penetración serán realizadas sobre sistemas de información más no sobre la infraestructura en la que se encuentran, teniendo en cuenta que esto depende directamente de los clientes, pues las pruebas se realizaran bajo ambientes de pruebas simulados. Las pruebas de penetración se dividen en algunas fases.

2.1.12.1 Fase de Reconocimiento: Ésta fase es en donde se establece el objetivo a atacar, se conoce más sobre el objetivo mediante la búsqueda de posibles vulnerabilidades que pueda tener. En el caso de las aplicaciones web, puede ser conocer más acerca de los framework que se usaron en los aplicativos, metodología de desarrollo usada, arquitectura y así conocer cuáles son las posibles debilidades que estas tengan.

2.1.12.2 Fase de Exploración: Ésta fase es en la cual se usa la información obtenida en la fase de reconocimiento con el fin de detectar debilidades sobre lo investigado, esto se toma como un tipo de ataque activo y se le podría llamar un pre ataque.

2.1.12.3 Fase de de evaluación y enumeración: Ésta fase es la encargada de obtener y recopilar información vital, tal como puertos abiertos, servicios, usuarios de red que estén disponibles en el sistema y que permitan un ataque que afecte al sistema de información o al servidor.

2.1.12.4 Fase de Informes: Ésta fase es en la cual se registran las vulnerabilidades encontradas en nuestro test de penetración y sobre estas se generan recomendaciones a tener en cuenta para evitar posibles ataques externos que puedan afectar, la disponibilidad, integridad y confidencialidad de los activos de la empresa.

2.2 MARCO HISTÓRICO

La seguridad de la información ha sido tenida en cuenta desde nuestros antepasados, pasando por uno de los primeros métodos de cifrado desarrollado por los griegos llamado “Escítala”, la cual hacía un cifrado por trasposición, esto con el fin de poder esconder información que era importante en ese entonces para ellos. Por ende no es incorrecto pensar que la seguridad de la información ha estado

inmersa siempre en el desarrollo de la civilización. Después nos encontramos por ejemplo con la creación de sociedades de hackers salidos del MIT (Instituto tecnológico de Masachussets), en donde llegaron los primeros computadores que ocupaban casi una habitación, en "1960: Los hackers originales utilizaron los primeros mainframes del MIT para desarrollar habilidades y explorar el potencial de la informática. En esa época, hacker era un término elogioso para los usuarios con un conocimiento exponencial de los ordenadores"[22].

En Colombia la seguridad de la información es un tema que se está viendo como uno de los pilares importante dentro de la sociedad, pues cada día vemos la creación de nuevas leyes y decretos que protegen los datos personales y la información confidencial como tal. De igual modo como se ha visto el avance de la tecnología los tipos de ataques realizados han avanzado también, por ejemplo "Según datos de la compañía Andina Digital Security en el último año cerca de 10 millones de colombianos han sido víctimas de algún delito informático"[23]. Y adicional a que no todas las personas denuncian este tipo de eventos, solo un bajo porcentaje de los afectados.

Por otro lado es importante tener en cuenta que la seguridad en los sistemas de información es vital, pues en estos puede residir información confidencial o importante de personas, procesos, secretos etc. Por ende la seguridad en las aplicaciones web es de vital importancia y por esto la evolución de las mismas a través de los lenguajes de programación y arquitectura de software apoyada con la infraestructura en donde se encuentran los sistemas de información instalados, esto con el fin de hacer mucho más seguro el resguardo de la información. Sin embargo en el correr de los años se ha visto como no solo es importante tener una buena infraestructura sino que también es de vital importancia que las fábricas de software tengan unas mejores prácticas en el desarrollo de los sistemas de información, pues un desarrollo mal realizado y sin metodología se puede volver una puerta para que se realicen ataques a los sistemas de información. Según Microsoft para tener en cuenta en los desarrollos la seguridad nos dan una lista de requisitos básicos, "La lista siguiente proporciona pautas de seguridad mínima que se aplican a todas las

[22] PUENTE, David. Un viaje en la historia del hacking. Magazine for software developers, programmers and designers [online], [Citado: 8 de Julio de 2009], Pag. 2.

[23]El espectador. Delitos Informáticos: Diez millones de colombianos, víctimas de delitos informáticos en el último año. En: El espectador [En línea]. Disponible en: <http://www.elespectador.com/tecnologia/diez-millones-de-colombianos-victimas-de-delitos-inform-articulo-442538> [Citado en: 26 de Agosto de 2013]

aplicaciones Web y que se deberían seguir: (Ejecutar aplicaciones con privilegios mínimos, Conocer a los usuarios, Protegerse contra entradas malintencionadas)"[24].

2.3 MARCO LEGAL

Este capítulo contemplará la parte normativa que respecta a los derechos de autor y a protección d la información, que es regulado por el código penal colombiano.

CAPITULO I.[25] DE LOS ATENTADOS CONTRA LA CONFIDENCIALIDAD, LA INTEGRIDAD Y LA DISPONIBILIDAD DE LOS DATOS Y DE LOS SISTEMAS INFORMÁTICOS.

ARTÍCULO 269A. ACCESO ABUSIVO A UN SISTEMA INFORMÁTICO.
ARTÍCULO 269B. OBSTACULIZACIÓN ILEGÍTIMA DE SISTEMA INFORMÁTICO
ARTÍCULO 269C. INTERCEPTACIÓN DE DATOS INFORMÁTICOS.
ARTÍCULO 269D. DAÑO INFORMÁTICO.
ARTÍCULO 269E. USO DE SOFTWARE MALICIOSO.
ARTÍCULO 269F. VIOLACIÓN DE DATOS PERSONALES.

CAPITULO II. DE LOS ATENTADOS INFORMÁTICOS Y OTRAS INFRACCIONES.

ARTÍCULO 269I. HURTO POR MEDIOS INFORMÁTICOS Y SEMEJANTES.
ARTÍCULO 269J. TRANSFERENCIA NO CONSENTIDA DE ACTIVOS.

[24] MSDN. Procedimientos de seguridad básicos para aplicaciones [en línea]. <http://msdn.microsoft.com/es-es/library/zdh19h94%28v=vs.100%29.aspx> [Citado en 2013].

[25] SECRETARIA DEL SENADO. CODIGO PENAL [ONLINE]. [Colombia]: 2011. Disponible en Internet: <URL: http://www.secretariasenado.gov.co/senado/basedoc/ley/2009/ley_1273_2009.html>

3. DISEÑO METODOLÓGICO

El método de investigación que es usado dentro de este proyecto es deductivo, pues la finalidad del proyecto es reducir la cantidad de vulnerabilidades dentro de los sistemas de información que pudieran ser explotadas en caso de ser encontrada alguna, a partir de un análisis inicial mediante un procedimiento de pruebas de penetración que es propuesto con ésta finalidad y mediante el cual se generan recomendaciones a seguir para disminuir el riego de ataques. La línea de investigación que será tenida en cuenta en este proyecto será la de técnicas y detección.

La realización de este proyecto fue definido del siguiente modo, se realizará en tres fases, en la primer fase se realizará la definición del proyecto como tal, los objetivos, la justificación, el problema, y demás. La segunda fase contendrá la definición del sistema de información con el cual se realizará la prueba piloto del procedimiento que se propondrá, se creará el procedimiento, se escogerán las herramientas con las cuales se realizarán las pruebas de penetración y posterior a esto se realizará la ejecución del procedimiento propuesto y se mostrarán resultados. A partir de la segunda parte se interpretarán los resultados y se generaran recomendaciones.

Mediante la consulta e investigación sobre cuál sería la mejor metodología para generar el procedimiento de pentest, nos encontramos con dos metodologías que son proyectos open source, OSSTMM (Open SourceSecutiry testing Manual) y OWASP (Open Web Applciation Security Project). De dichas metodologías se ha seleccionado la metodología OWASP basándonos en los siguiente:

OSSTMM, es una metodología que se encarga de auditar la seguridad en las empresas teniendo en cuenta 6 grandes módulos que se enfocan en diferentes objetivos cada uno, como lo son:

- Seguridad de la información, asociado a la presencia de información de la empresa en internet.
- Seguridad en los procesos, asociada a la ingeniería social.
- Seguridad en tecnologías de internet, asociado a métodos de contención y demás como los son firewalls, ids, router testing, DOS, etc.
- Seguridad de las comunicaciones, asociado a modems, fax, etc.
- Seguridad wireless, asociado a comunicaciones por Bluetooth, infrarrojos y protocolos 802.*. etc.

- Seguridad física, asociado a controles de acceso, perímetro de seguridad, alarmas, etc.

OWASP, esta metodología se encarga de auditar aplicaciones web, mediante un catálogo de 66 controles a tener en cuenta para revisar toda la aplicación, esta metodología se hace eficiente con respecto al pentest sobre aplicaciones web al tener dentro de la misma algunos consejos y explicaciones útiles cuando no se tiene mucha experiencia en el tema de pentestig, tales como el funcionamiento de algunos ataques, herramientas que se pueden usar etc. Esta metodología está dividida en 9 dominios a tener en cuenta al momento de auditar una aplicación web.

- Recopilación de información.
- Test de gestión de la configuración.
- Test de autenticación.
- Test de autorización.
- Test de gestión de sesiones.
- Test de autorización.
- Test de validación de datos de entrada.
- DOS testing.
- Test de web services.
- Test AJAX.

Por ende, y acorde a lo anterior, la metodología OSSTMM ve de un modo holístico la seguridad, sin embargo y para efectos de este proyecto no es lo más conveniente, teniendo en cuenta que, el procedimiento que se va a generar será un procedimiento corto, ágil y de bajo costo para que las fábricas de software lo puedan ejecutar en cualquier momento sin perjudicar costos ni tiempo sobre el Proyecto en gran medida. Adicional a esto también hay que tener presente que la metodología OWASP está enfocada a aplicaciones web específicamente como se explico anteriormente.

Por ende la metodología OWASP es la más acorde al objetivo de este Proyecto. Se aclara que no todo lo descrito de la metodología será ejecutado, pues la idea de este Proyecto es generar un procedimiento liviano que no produzca grandes impactos en tiempo y dinero dentro del ciclo de vida del software en la etapa de pruebas, en conclusión se tomarán algunos controles de la metodología para implementar el procedimiento, tomando como base que el procedimiento debe ser un procedimiento liviano y con el cual se pueda probar y verificar la mayor superficie de ataque que pueda tener la aplicación y pueda ser usada por usuarios malintencionados, dicho lo anterior se seleccionaron los siguientes controles a

desarrollar en el procedimiento cubriendo así el top 10 de las amenazas más comunes publicado por el OWASP (Ver Cuadro 2), del siguiente modo:

1. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Inyección de código malicioso, redireccionamientos inválido a otros sitios de un portal, objetos referenciados directamente), mediante los controles (OWASP-IG-003 “Identificar puntos de entrada”, OWASP-DV-001 ”Reflected XSS” y OWASP-DV-005 ”SQL injection”) descritos en este procedimiento.

2. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Configuración incorrecta), mediante los controles (OWASP-IG-004 “Testing for web applciation fingerprint”) descritos en este procedimiento.

3. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Información sensitiva expuesta), mediante los controles (OWASP-CM-005 “Testing for file extensions handling” y OWASP-IG-001”Spiders and crawlers”) descritos en este procedimiento.

4. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Usar componentes con fallas conocidas), mediante los controles (OWASP-IG-004 “Testing for web applciation fingerprint” y OWASP-CM-001 “SSL/TLS Testing (SSL Version)”) descritos en este procedimiento.

5. Adicionalmente en el procedimiento se ejecuta un análisis de vulnerabilidades el cual es usado como refuerzo a las validaciones anteriores y el cual da chance para realizar validaciones manuales en caso de encontrarse vulnerabilidades nuevas.

Las validaciones del procedimiento se realizaran al aplicativo, más no a la infraestructura final, pues esta es siempre provista por el usuario al ser autónomos en la selección donde estarán alojados sus sistemas de información, ni temas dependientes a la infraestructura de los clientes.

Cuadro 5. Controles seleccionados

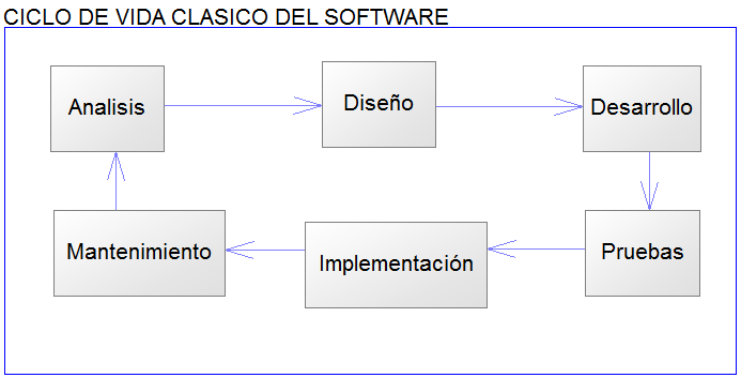
Category	Número de referencia	Test Name	Vulnerability

Recopilación de información (Information gathering)	OWASP-IG-001	Spiders, Robots and Crawlers -	N.A.
	OWASP-IG-003	Identify application entry points	N.A.
	OWASP-IG-004	Testing for Web Application Fingerprint	N.A.
Gestion de configuración (Configuration Management Testing)	OWASP-CM-001	SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert. Validity)	SSL Weakness
	OWASP-CM-005	Testing for File Extensions Handling	File extensions handling
Validación de datos de entrada (Data Valdiation Testing)	OWASP-DV-001	Testing for Reflected Cross Site Scripting	Reflected XSS
	OWASP-DV-005	SQL Injection	SQL Injection

Fuente. De los autores.

Ahora bien, a partir de los controles seleccionados sobre los cuales el procedimiento fue basado, se ha generado un diagrama de flujo, en donde se verán las actividades que contendrá el procedimiento y al igual se mostrará el ciclo de vida clásico del software y donde el procedimiento será ubicado (Ver Figura 3).

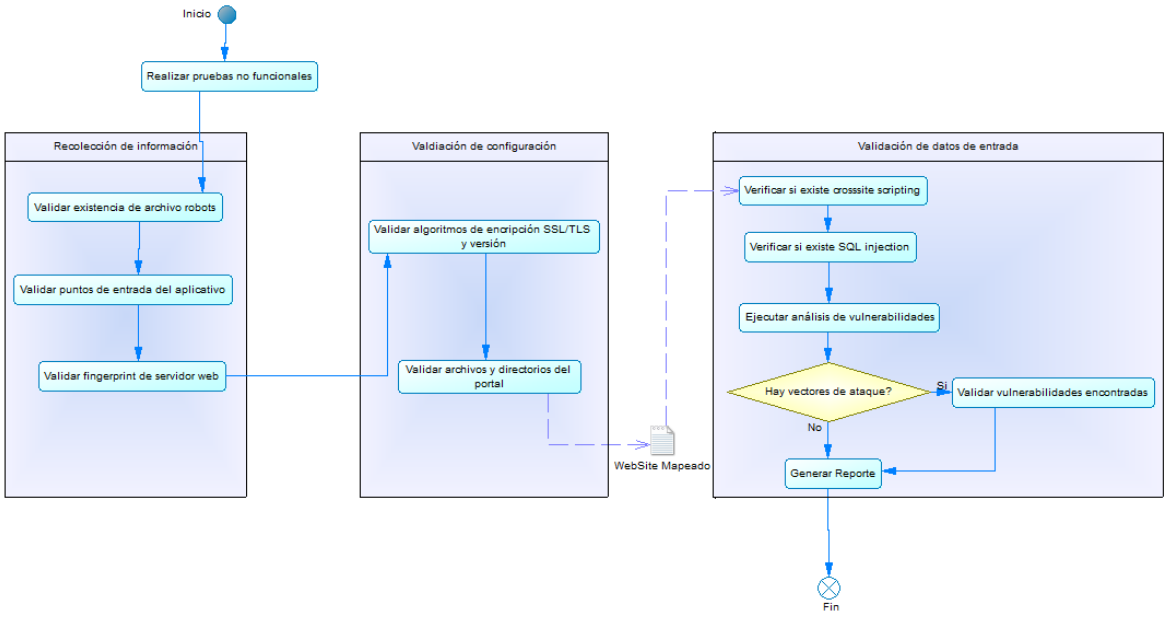
Figura 3 Ciclo de vida clásico del software



Fuente. De los autores.

Como se puede apreciar en la figura 3, el ciclo de vida clásico del software contiene una etapa de pruebas, dentro de dicha etapa estará ubicado el proceso de prueba de penetración, y estará dentro del tipo de prueba no funcional, a continuación se mostrará por medio de un diagrama de flujo las actividades dentro del procedimiento (Ver Figura 4).

Figura 4 Flujo de trabajo del procedimiento



Fuente. De los autores.

Se debe tener presente que el flujo anterior muestra el procedimiento completo, sin embargo dentro de cada una de las actividades se describirán a continuación tanto

las entradas y las salidas de cada uno, teniendo en cuenta que; No necesariamente las actividades generan entradas para la siguiente actividad, pues algunas son actividades separadas con un objetivo de validación específico. Para la validación del procedimiento se ha generado un documento anexo llamado "Procedimiento_Pentest_v1.0.docx", el cual fue realizado basándonos en la estructura de un archivo de ejemplo (Sample Procedure QSP 5.7.1) de la norma ISO 17025, la cual especifica los requisitos generales para realizar test o muestreos[26].

Adicionalmente se generó un archivo anexo con la ejecución del procedimiento al sistema de información seleccionado, dicho archivo anexo se llama "Ejecución_Procedimiento.docx".

[26] ISO/IEC 17025:2005. General requirements for the competence [en línea]. < http://www.iso.org/iso/catalogue_detail.htm?csnumber=39883>. [Citado en 2014]

4. CONCLUSIONES

Por medio de la elaboración del procedimiento se evidencia que si es necesario que las empresas dedicadas a desarrollo de software realicen sobre los sistemas de información pruebas de penetración, para así minimizar la superficie de ataque que puedan aprovechar los hackers en los ambientes productivos.

Se pudo validar que los sitios web se encuentran expuestos a gran variedad de vulnerabilidades, sin embargo, una de las principales partes corresponde; a que tanta información puede suministrar un sitio web, en el caso de estudio, se pudo validar que información importante salió directamente de archivos que se encontraban publicados con el sitio y a los cuales no se debería tener acceso.

Se logró crear un procedimiento de bajo costo y que puede ser usado por las empresas de software al haber sido generado con herramientas Open source en un 90%, por ende más accesible para las mismas.

Se logró crear un procedimiento que no generará impactos fuertes en tiempo para las empresas, teniendo en cuenta que el procedimiento ejecutado tardó 9 horas 10 minutos en ser ejecutado completamente y cubriendo así las vulnerabilidades expuestas por la metodología OWASP en su top 10.

El tiempo de ejecución de este tipo de test puede variar dependiendo del tamaño del sitio web al cual se le vayan a realizar las pruebas y el servidor en el cual el sitio web este hospedado.

Se concluyó que la seguridad en los sitios web es de suma importancia, sin embargo, dicha seguridad en los despliegues debe estar acompañada de una muy buena configuración en los ambientes productivos, refiriéndonos no solo al software sino al hardware y configuración de la red como tal dentro de las empresas o donde se vayan a hospedar los sitios que son desarrollados.

Se recomienda que para los CMS (administradores de gestión de contenidos por sus siglas en inglés), en nuestro caso Drupal, mantener al día las actualizaciones, y plugins de seguridad que son liberados para así evitar posibles vulnerabilidades.

Ahora, para desarrollos que sean realizados a la medida y en diferentes lenguajes de programación es necesario que las librerías que serán usadas en el desarrollo

estén actualizadas y no contengan vulnerabilidades conocida, para así evitar posibles vectores de ataque sobre los aplicativos.

5. BIBLIOGRAFÍA

MITNICK, Kevin y SIMON, William. El arte de la intrusión, como ser un hacker o evitarlos. México, Alfaomega Grupo editor, S.A de C.V., 2007, 371p.

TORI. Carlos. HACKING ETICO. Rosario, Argentina. [s.n.], 2008. 328p

WILHELM. Thomas. Professional Penetration Testing. Jordan Hill, Oxford. Syngress, 2010. 495p

BROAD, James y BINDER, Andrew. Hacking with kali, Practical Penetration Tecniques. Waltham, Syngress, 2014, 223p.

BECHIMOL, Daniel. Hacking desde cero. Buenos Aires, Fox Andina, 2011, 192p.

SHAKEED, Ali y TEDI, Heriyano. Assuring Security by penetration testing. Birmingham, UK, Pack Publishing Ltd, 2011, 392p.

GRAVES, Kimberly, Oficial Certified Ethical Hacker Review guide, Indianapolis, Indiana, Sybex, 2007, 167p.

6. ANEXOS

ANEXO A. Procedimiento

1 Propósito del documento

El propósito de este documento es describir el procedimiento de pruebas de penetración que se propone, teniendo en cuenta, las entradas y salidas de cada punto que será tenido en cuenta dentro del procedimiento así como la ejecución de cada uno.

2 Alcance

El procedimiento que se describe en este documento fue realizado basándonos en la metodología OWASP, con la cual se busca minimizar la superficie de ataque que pueda tener un aplicativo web desarrollado por la fábricas de software, los controles que fueron seleccionados se tomaron a partir de la validación del top 10 de las amenazas más comunes publicado por el OWASP, cubriendo del siguiente modo dichas amenazas:

6. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Inyección de código malicioso, redireccionamientos inválido a otros sitios de un portal, objetos referenciados directamente), mediante los controles (OWASP-IG-003 “Identificar puntos de entrada”, OWASP-DV-001 ”Reflected XSS” y OWASP-DV-005 ”SQL injection”) descritos en este procedimiento.
7. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Configuración incorrecta), mediante los controles (OWASP-IG-004 “Testing for web applciation fingerprint”) descritos en este procedimiento.
8. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Información sensitiva expuesta), mediante los controles (OWASP-CM-005 “Testing for file extensions handling” y OWASP-IG-001”Spiders and crawlers”) descritos en este procedimiento.
9. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Usar componentes con fallas conocidas), mediante los controles (OWASP-IG-004 “Testing for web applciation fingerprint” y OWASP-CM-001 “SSL/TLS Testing (SSL Version)”) descritos en este procedimiento.
10. Adicionalmente en el procedimiento se ejecuta un análisis de vulnerabilidades el cual será usado como refuerzo a las validaciones anteriores y el cual da chance para realizar validaciones manuales.

3 Definiciones

A continuación se presentan algunos términos utilizados en este documento que hacen relación al proceso de pruebas.

Términos	Definición
Pentest	Es como comúnmente se denomina a los (Test de penetración) y son en conjunto la forma de denominar a una serie de técnicas utilizadas para evaluar la seguridad de redes, sistemas de computación y aplicaciones involucradas en los mismos
Entregable	Resultado tangible (documento, formato, software, etc.) de una tarea completada.
Tester	Analista de pruebas que ejecuta las pruebas. *<Indicar nombres de los tester implicados en la ejecución>*

4 Herramientas a usar

- **Kali de Linux:** Es un Proyecto open source que es mantenido y fue fundado por Offensive and security, un proveedor de clase mundial de entrenamiento en seguridad informática y servicios de test de penetración. Kali de linux es una distribución GNU/Debian diseñada principalmente para auditoria y seguridad informática en general.

- **Web Scarab:** Webscarab es un marco de trabajo para analizar aplicaciones web que se comunica usando los protocolos HTTP y HTTPS. Está escrito en Java, por lo que es portable a muchas plataformas. WebScarab tiene muchos modos de operación, implementados por varios plugins. Su uso más común es operar WebScarab 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.

- **Nmap:** Es un programa de código abierto, que se usa para escanear puertos, servicios y descubrir servidores en una red informática.

- **Zenmap:** Es el GUI oficial de seguridad de Nmap, se trata de una plataforma multi (Linux, Windos, Mac Os entre otros), el cual tiene como objetivo tener a disposición las mismas funcionalidades de nmap, pero de un modo gráfico y más sencillo de utilizar para principiantes y expertos en nmap.

- **Openssl:** Es una herramienta de criptografía que implementa SSL y TLS protocolos de red y los estándares requeridos por estos. Este software es

desarrollado y mantenido por una comunidad abierta de voluntario con gran conocimiento en seguridad de la información.

- **Accunetix:** Accunetix es una herramienta que puede ser capaz de escanear sitios web en busca de fallos de seguridad que puedan poner en peligro la integridad de un sitio web publicado en internet.

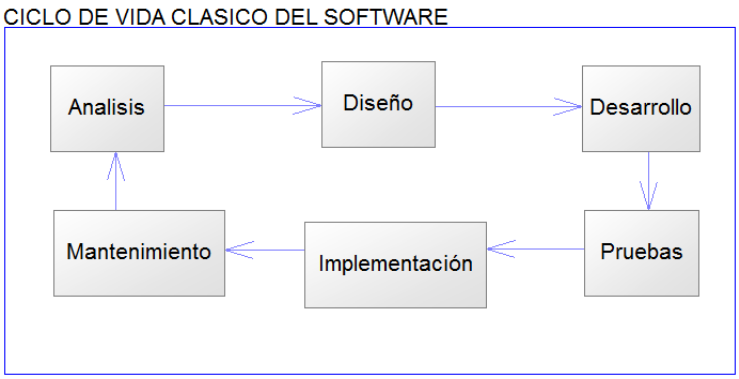
- **SQLMAP:** Sqlmap es una herramienta de pentesting open source que automatiza la el proceso de detección y explotación de fallas de sql injection.

- **Nikto:** Nikto es un web server scanner que realiza test contra aplicaciones web, usando múltiples validaciones varios items, como lo son archivos y directorios potencialmente peligrosos, validación de actualizaciones entre otros.

5 Flujo de trabajo del procedimiento

A continuación se mostrará el flujo de trabajo que el procedimiento tiene y se mostrará en que etapa del ciclo de vida clásico de software está inmerso el procedimiento para su ejecución (Ver Figura 1).

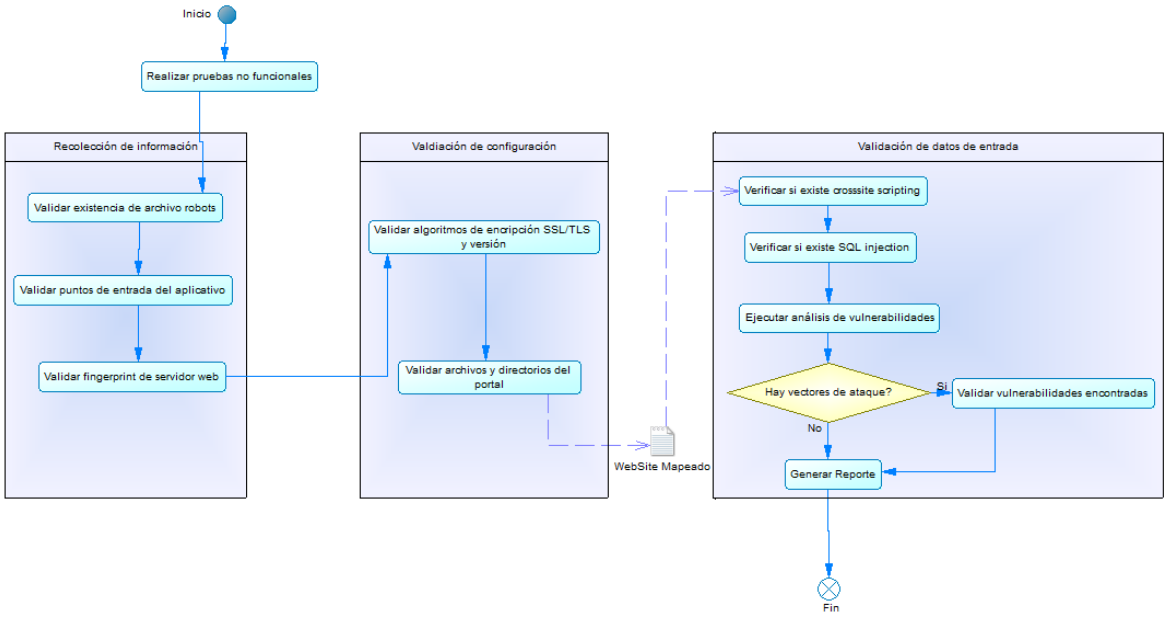
FIGURA 1. Ciclo de vida clásico del software



Fuente. De autores

Como se puede apreciar en la figura 1, el ciclo de vida clásico del software contiene una etapa de pruebas, dentro de dicha etapa estará ubicado el proceso de prueba de penetración, y estará dentro del tipo de prueba no funcional, a continuación se mostrará por medio de un diagrama de flujo las actividades dentro del procedimiento (Ver Figura 2).

FIGURA 2. Flujo de trabajo del procedimiento



Fuente. De autores

Se debe tener presente que el flujo anterior muestra el procedimiento completo, sin embargo dentro de cada una de las actividades se describirán a continuación tanto las entradas y las salidas de cada uno, teniendo en cuenta que; No necesariamente las actividades generan entradas para la siguiente actividad, pues algunas son actividades separadas con un objetivo específico.

6 PROCEDIMIENTO

A continuación se describe el procedimiento propuesto a usar para la ejecución de pruebas de penetración, dentro de cada ítem se pueden apreciar las entradas, pasos de ejecución y posibles salidas de cada una de las actividades descritas anteriormente en el flujo de trabajo.

6.1 Validar existencia de archivo robots.txt (Control OWASP-IG-001)

Objetivo: Realizar la verificación de la existencia del archivo “Robots.txt”, con el cual se podrá validar si existen directorios dentro del portal que no se quieran ser indexados por los buscadores como google, bing etc.

Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación.

Actividades: En kali de linux, abra una terminal y ejecute el siguiente comando “wget http://www.ejemplo.com/robots.txt” y abra el archivo descargado usando el

comando “vi Robots.txt” ubicado en el path en el cual guardo el archivo o abra la url en un navegador web directamente (http://www.ejemplo.com/robots.txt).

Salidas: En ésta validación solamente habrán dos posibles salidas, una dando como resultado positivo el hallazgo del archivo que contendrá una listado de directorios con disallow (Para directorios que el administrador del sitio no quiere que los buscadores indexen o allow, para sobre los cuales los buscadores podrán realizar indexaciones) y un resultado negativo dando como respuesta que el archivo no existe.

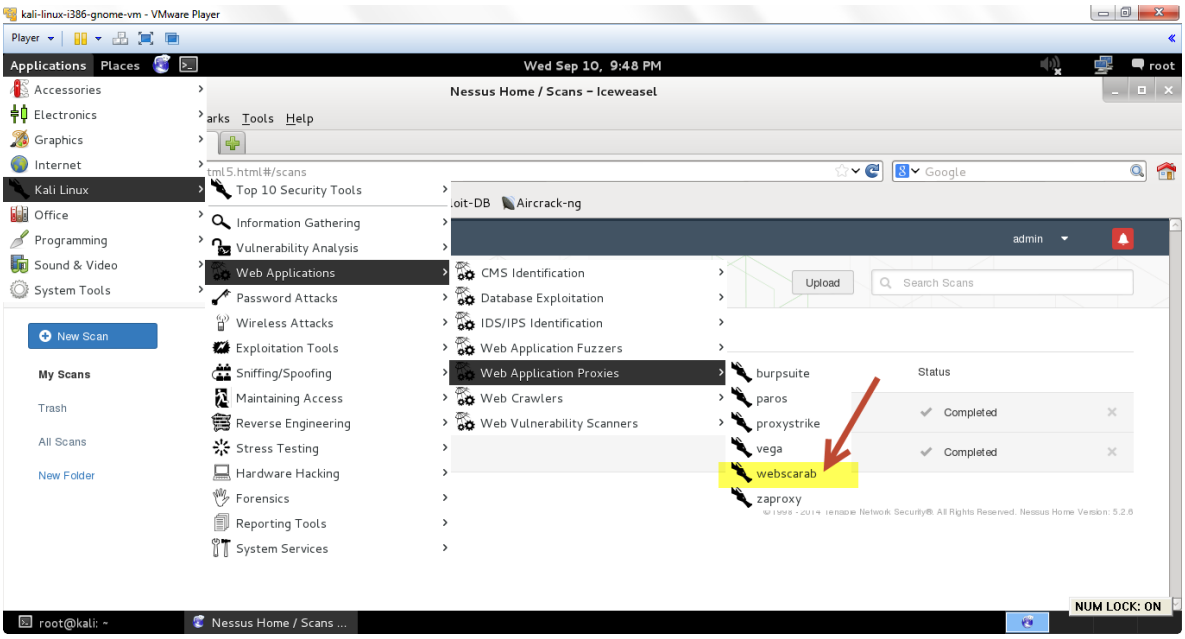
6.2 Validar putos de entrada del portal (Control OWASP-IG-003)

Objetivo: Verificar dentro del portal a analizar cuáles podrían ser posibles puntos de ataque, como lo pueden ser formularios dentro del portal (ingreso, autenticación, registro etc.), tratando de validar así una superficie de ataque con posibles campos no validados correctamente, para realizar esta validación se tendrán en cuenta los métodos GET y POST que usa la aplicación en sus operaciones.

Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación y los formularios a los cuales se les realizará la interceptación de parámetros.

Actividades: Este punto se realizara con la herramienta llamada webscarab, la cual servirá como proxy para realizar la interceptación de datos que usa el portal enviados y recibidos por los métodos GET y POST, para este punto debemos enfocarnos en los formularios que tenga el portal.
Para usar la herramienta abra la maquia virtual Kali de Linux, y abra el aplicativo webscarab mediante el menú que se presenta (Ver Figura 3).

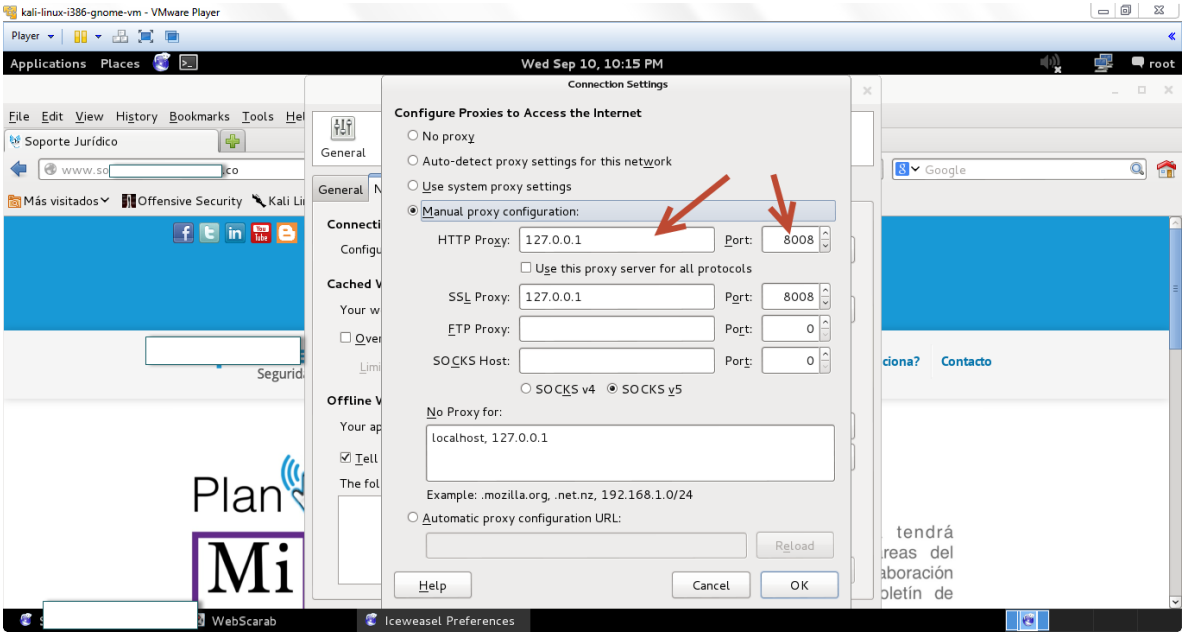
FIGURA 3 Abrir WebScarab desde Kali Linux



Fuente: De Autores

Después de haber abierto el aplicativo webscarab en el navegador web ingrese al portal por medio de la URL, estando dentro del navegador, modifique las preferencias e ingrese en configuración manual del proxy en http proxy “127.0.0.1” y puerto 8008, dicha configuración es necesaria y es la dada por webscarab (Ver Figura 4).

FIGURA 4 Configuración manual del proxy (Webscarab)



Fuente: De Autores

Después de haber configurado esto en el navegador, web scarab ya queda configurado como proxy, por ende lo que se debe realizar posterior a esta configuración es realizar cualquier operación (como consulta o ingreso de información a cualquier formulario).

Salidas: La salida de esta importante operación es que en webscarab aparecerán los métodos GET y POST usados por la o las operación realizadas por el usuario mostrando los parámetros usados y que se enviaron al servidor.

6.3 Validar fingerprint del servidor de aplicaciones (Control OWASP-IG-004)

Objetivo: Verificar primero que todo si la configuración del servidor permite al tester obtener la versión del servidor de aplicaciones en el cual el aplicativo está instalado y a partir de esto validar si la versión del servidor que se está usando está actualizado y tiene alguna vulnerabilidad conocida.

Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación y algunos comandos necesarios a usar en el aplicativo zenmap que serán explicados en las actividades.

Actividades: Para esta validación usaremos el software llamado zenmap, el cual puede ser instalado en nuestra maquina kali de Linux, en caso que no se encuentre instalada puede instalarla ejecutando los siguientes comandos desde una terminal de Linux (**sudo apt-get install zenmap**), ahora para ejecutar el zenmap que se acabo de instalar ejecute los siguientes comandos (**sudo zenmap**) y posterior a la

ejecución abrirá zenmap automáticamente. Cuando se tenga zenmap abierto, debe ingresar la siguiente información, en target (ingresar el nombre del host a escanear), en profile seleccione (Intense), dicho perfil ejecutará el escaneo del siguiente modo (–T4, este comando equivale a la velocidad del escaneo, va de 0 a 5, donde el mayor valor aumentará la velocidad de escaneo), (-v, este comando traerá la mayor información que se pueda, en caso que se quiera conocer más información se deben aumenta la cantidad de 'v' en el campo command) y (-A, este comando ejecutara la detección de versiones, sistema operativo y traza de ruta).

Salidas: En ésta validación se obtendrá como salida la versión del servidor de aplicaciones en donde se encuentra instalada la aplicación.

6.4 Validar métodos de encripción (Control OWASP-CM-001)

Objetivo: Verificar si en el servidor existen métodos de encripción seguros, y adicional a esto verificar si estos están actualizados o no.

Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación y algunos comandos necesarios a usar en el aplicativo nmap y openssl que serán explicados en las actividades.

Actividades: Para validar este punto tomaremos como herramientas y punto de partida el uso de la herramienta nmap y openssl, por medio de la herramienta nmap, validaremos si existe un servicio SSL activo, esto mediante el comando (nmap –F –sV www.host.com), donde el comando (-F, corresponde a la velocidad del scaneo, en este caso F corresponde a fast), y el comando (-sV, prueba puertos abiertos para definir servicios y versiones de los mismos.), para nuestro caso validaremos si existe servicio SSL activo, si existe el servicio el sistema mostrará en los resultados el puerto por el cual el servicio está siendo ejecutado.

Posterior a la validación anterior usaremos el aplicativo openssl para verificar si efectivamente en el puerto que arrojo la validación del punto anterior corre algún servicio SSL y así validar la versión del mismo.

Validaremos la versión de SSL y TLS usados, mediante el uso de la herramienta Openssl. Ejecutaremos el siguiente comando en una terminal Linux (openssl s_client -no_tls1 -no_ssl2 -connect www.host.co:443), si el resultado es exitoso el sistema nos mostrará la información del servidor como, versión de ssl y tls, certificado del servidor y algoritmo de encripción usado. El comando (s_client, es usado para crear un cliente SSL / TLS genérico para conectarnos a un host remoto), los comando (-no_tls1 y –no_ssl2, son usados para conectarnos al host remoto omitiendo los protocolos nombrados, en este caso omitiría la conexión con sslv2 y tlsv1 y trataría de conectar con el servidor por medio de sslv3), y por último el comando (-connect www.host.co:443, es usado para especificar a qué servidor remoto nos queremos conectar y por medio de que puerto).

Salidas: En ésta validación se obtendrá como dada por las dos herramientas, por un lado los servicios en ejecución y puerto, y por otro lado obtendremos las versiones de encripción SSL / TLS y si el servidor usa certificados de seguridad.

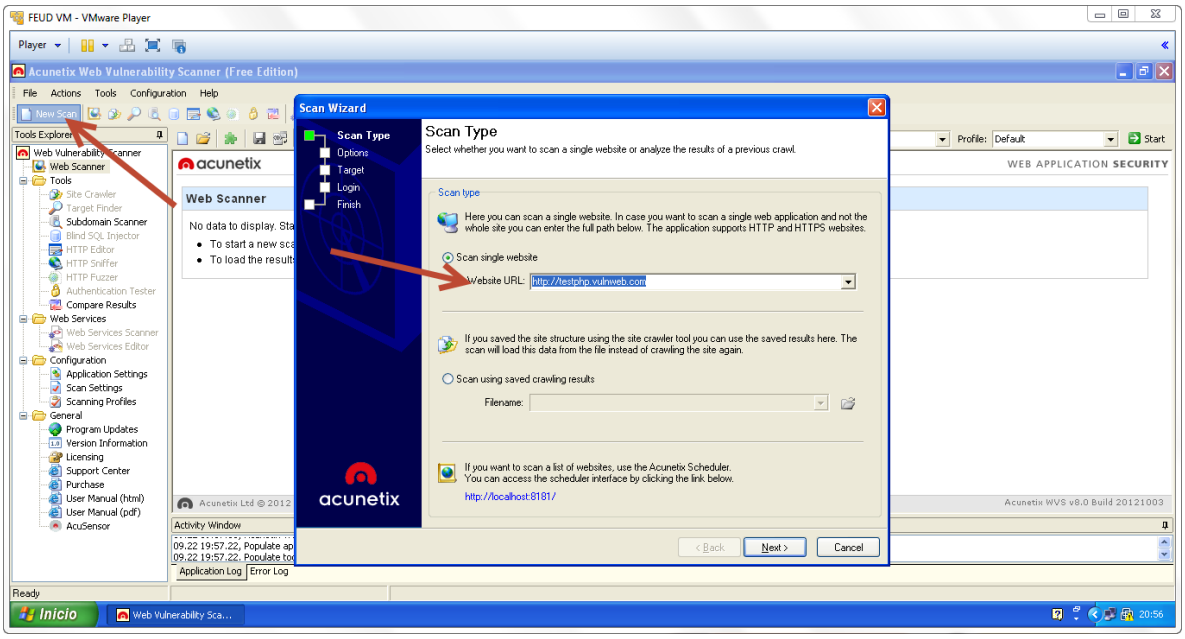
6.5 Validar configuración y manejo de archivos (Control OWASP-CM-005)

Objetivo: Verificar si el portal está manejando bien los archivos, evitando que esté expuesto al usuario información sensible o archivos de configuración.

Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación y la versión de accunetix free version instalada, adicionalmente se aclara que para este punto se puede usar también el aplicativo llamado DIRBUSTER.

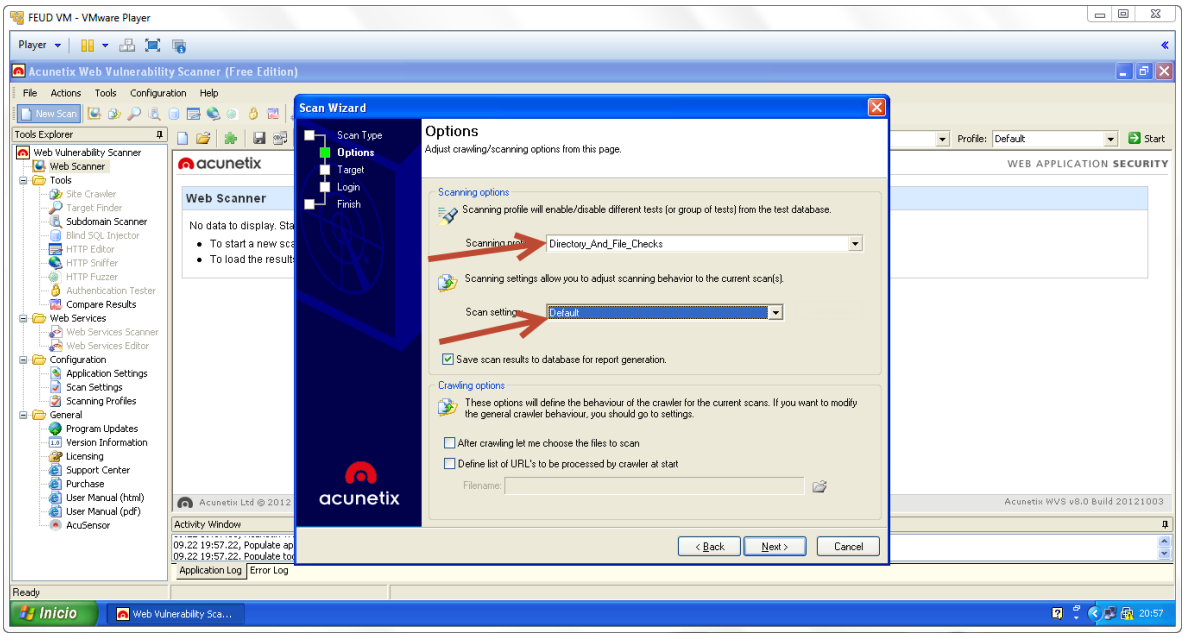
Actividades: Para realizar ésta validación usaremos un software que se llama Accunetix (Free version), el cual realiza un escaneo completo en el sitio en búsqueda de los diferentes tipos de archivos que este sitio contenga. Este software puede ser usado en sistemas operativos Windows. Para instalar el software dirigirse al siguiente sitio web (http://www.acunetix.com/vulnerability-scanner/download/), descargar e instalar la versión de prueba. Para configurar el escaneo, realizar los siguientes pasos después de haber instalado el software, iniciar el acunnetix y dar click en “new scan” (Figura 5), en la ventana que aparece en el campo llamado website URL coloque el nombre de dominio del sitio al cual le quiere realizar el escaneo y dar clic en siguiente, en la siguiente ventana, seleccione el tipo de escaneo en el campo llamado “scan profile” y en el campo scan settings “default” (Figura 6), al ser la versión de prueba es la única opción que se tendrá disponible y dar clic en siguiente, en la siguiente ventana saldrán los datos principales del host a escanear, dar clic en next en la ventana que aparece dar click nuevamente en next (Figura 7). Posterior a esto el sistema empezará a realizar el escaneo del sistema de información que se configuró mostrando la estructura que se encontró del portal.

FIGURA 5 Wizard iniciar escaneo



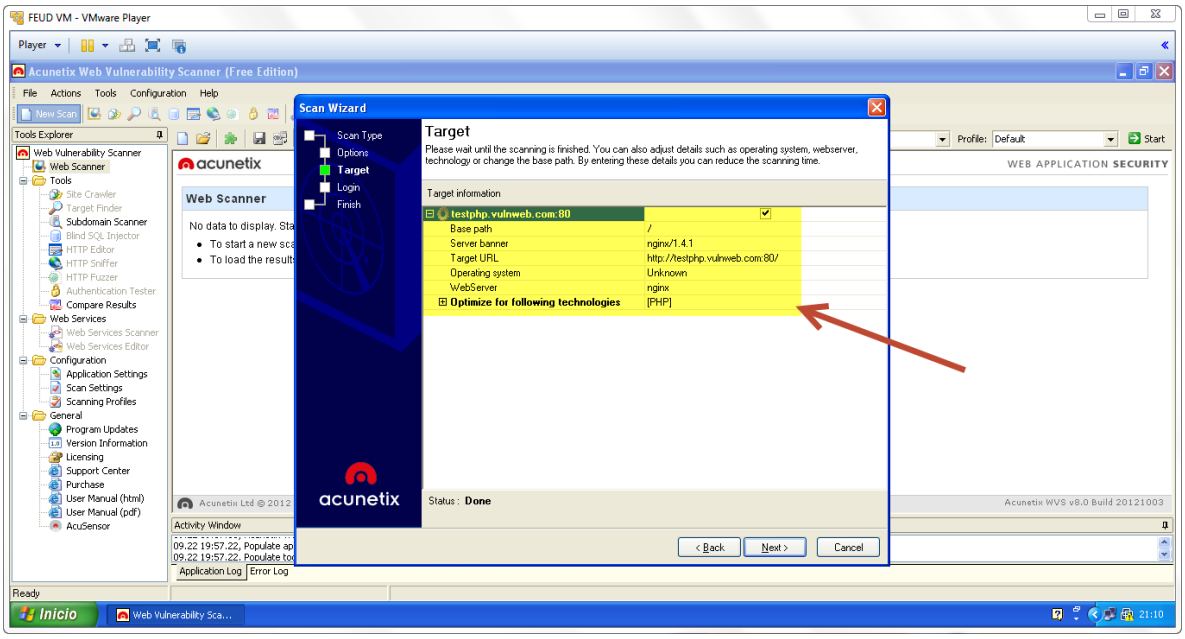
Fuente: De Autores

FIGURA 6 Wizard configurar perfil de escaneo



Fuente: De Autores

FIGURA 7 Finalizar configuración de escaneo



Fuente: De Autores

Salidas: Accunetix nos mostrará la estructura del sitio y los archivos que logró localizar que el portal tiene expuestos.

6.6 Validar existencia de inyección cross site scriptng y CRLF (Control OWASP-DV-001)

Objetivo: Verificar si de alguno de los puntos de entrada validados en el punto (6.2 Control OWASP-IG-003) se puede sacar ventaja utilizando la vulnerabilidad de XSS ó CRLF. Este tipo de prueba consiste en tratar de inyectar código javascript o html, por ejemplo en formularios, variables pasadas por la URL etc.

Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación y algunos de los puntos de entrada que se quieren probar, teniendo en cuenta que pueden o no ser vulnerables.

Actividades: Para esta prueba se debió haber realizado en la fase de recolección de información una navegación completa por el sitio, para así detectar los posibles punto de entrada, por ende a continuación se explicará cómo se validara este test usando la herramienta webscarab. Para iniciar, se debe configurar el webscarab como proxy en el navegador web, para que quede a la escucha de las transacciones realizadas tal como se explico en el punto “6.2” “Figura 4” de este procedimiento ya, configurado esto, en webscarab seleccionar la opción llamada (XSS/CRLF) y empiece a navegar dentro del portal, y a continuación webscarab iniciara listando las páginas y las respectivas variables en las cuales se podría presentar un posible ataque.

Nota: Se aclara que en este punto webscarab solamente listará las páginas que pueden que tener puntos de entrada para realizar un ataque de cross site scripting, sin embargo, no todos los listado por webscarab son vulnerabilidades efectivas, por ende la prueba de esto debe realizarse manualmente probando las URL listadas por webscarab.

Después de haber listado las variables webscarab le permite probar de una manera manual las url que posiblemente pueden tener ataque por XSS / CRLF, seleccionando de la lista la URL a probar, de click en el botón “Edit Test Settings” e ingresando allí el valor con el cual desea probar la posible vulnerabilidad, de click en ok y después para probar la vulnerabilidad de click en check, si despues de dar click en check la URL aparece en la grilla inferior, quiere decir que la vulnerabilidad existe, de lo contrario es porque no existe dicha vulnerabilidad en la URL seleccionada.

Salidas: Si los resultados de la prueba con webscarab son satisfactorios, el sistema mostrará al usuario las URL o los parámetros que pueden ser inyectados ya sea para la vulnerabilidad XSS o CRLF.

6.7 Validar existencia de SQL inyección (Control OWASP-DV-005)

Objetivo: Validar si existe dentro del portal verificado alguna vulnerabilidad asociada a Sql Injection, que permita a un atacante obtener información que está contenida en la base de datos y sea de uso exclusivo del aplicativo.

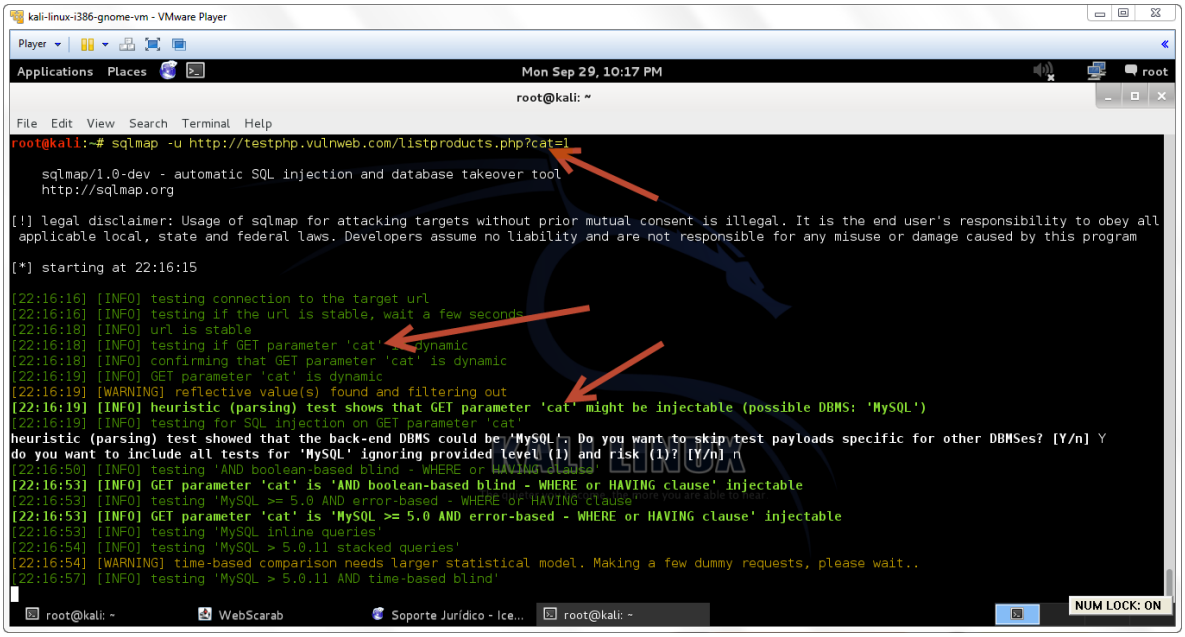
Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación y algún parámetro o URL de la cual se sospeche se pueda realizar un ataque.

a. **Actividades:** Para validar si en un portal existen vulnerabilidades de SQL injection se tienen dos posibilidades, se puede realizar una validación manual o una validación automática que puede realizar un escáner de vulnerabilidades web como lo es nikto o accunetix, sin embargo se debe tener presente que en caso que salieran vulnerabilidades listadas por los escáneres asociadas a Sql injection, estas deben ser validadas manualmente. A continuación se explicará cómo realizar dicha labor.

Para efectos de esta prueba se seleccionará la siguiente URL a la cual se le realizaran pruebas por medio del software SQLMAP, para usar este software, ya se debe tener ubicada la URL de la cual se sospecha o es vulnerable al ataque de SQL injection, en nuestro ejemplo usaremos la siguiente URL (http://testphp.vulnweb.com/listproducts.php?cat=1) con la cual se podrá ver el funcionamiento del software sobre una vulnerabilidad de Sql injection (Ver Figura 8).

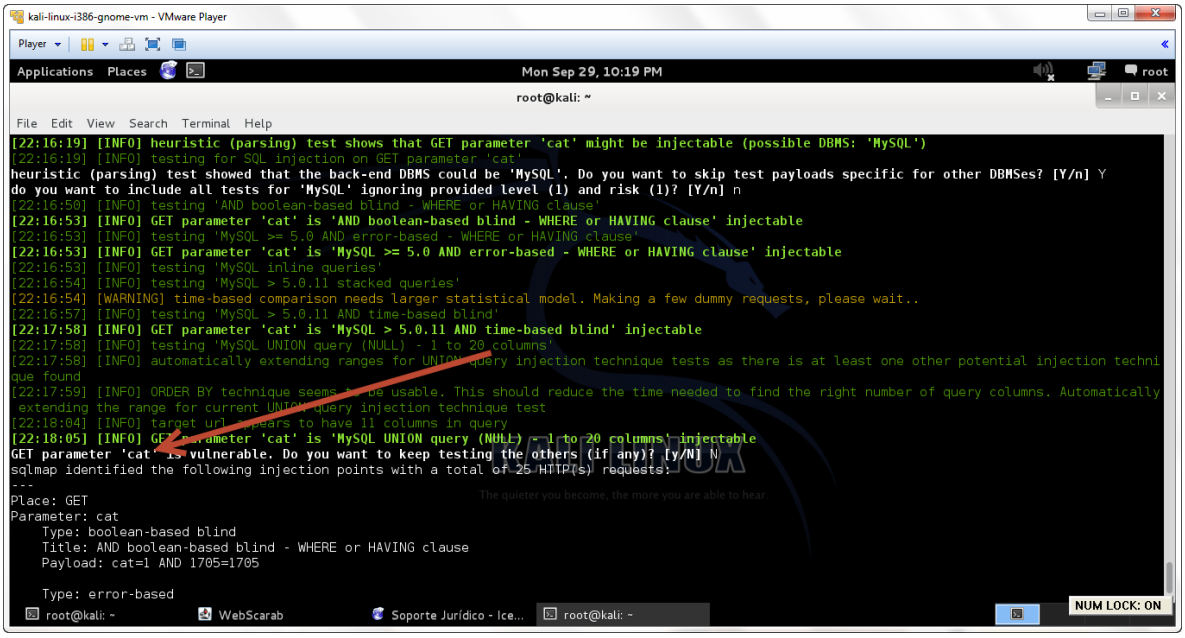
En una terminal de Linux dentro de kali ejecutar el siguiente comando “sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1”, con este comando se validará si la URL que se ingreso es o no vulnerable al ataque (Ver Figura 9).

FIGURA 8 Resultados de ejecución de comando SQLMAP



Fuente: De Autores

FIGURA 9 Confirmación de vulnerabilidad encontrada



Fuente: De Autores

Salidas: Si los resultados de la prueba son satisfactorios sobre la URL de la cual se sospechaba el sistema de información SQLMAP le permitirá consultar y obtener información de la URL tales como tablas, estructuras de bases de datos entre otros, sin embargo si la URL no resulta ser inyectable el sistema de información SQLMAP se lo hará saber.

6.8 Ejecutar análisis de vulnerabilidades Web

Objetivo: Verificar con un scanner de vulnerabilidades web si existen dentro del portal vulnerabilidades adicionales que no hayan sido verificadas.

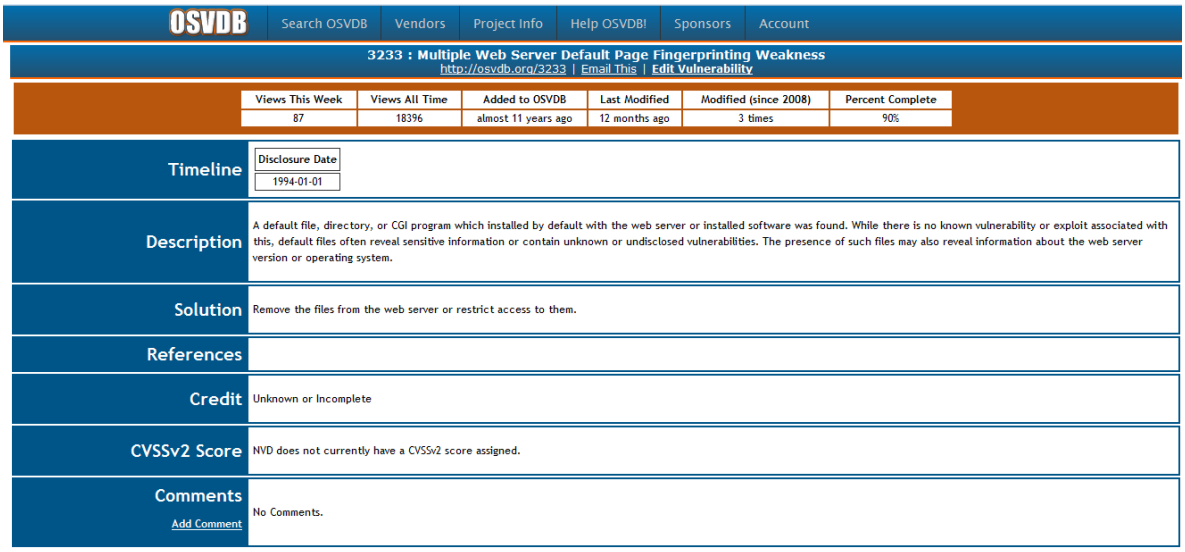
Entradas: Para realizar la validación de este punto es necesario tener claro el host al cual se le va a realizar la verificación.

Actividades: Para esta prueba es necesario usar un scanner de vulnerabilidades web llamado Nikto. Para realizar la ejecución del análisis abra una terminal Linux y ejecute el siguiente comando (nikto -host www.nombrehost.co) e inmediatamente Nikto iniciara a realizar el scan del sitio, hay que tener en cuenta que al enviar ningún parámetro adicional a Nikto mas que el Host, Nikto realizará todas las validaciones sobre el portal.

Salidas: Si los resultados de la prueba con Nikto son satisfactorias generará un listado de posibles problemas dentro del portal y el web server, añadiendo a este un

código del tipo de vulnerabilidad encontrada que es clasificada por la OSVDB (Open source vulnerability database). Por cada vulnerabilidad encontrada por Nikto este le asignará un código que puede ser consultado en la página de OSVDB el cual presentará por vulnerabilidad un informe al respecto como el que se verá a continuación (Ver Figura 10).

FIGURA 10 Informe de vulnerabilidad OSDVBD



Fuente: http://osvdb.org/3233

7 Revisiones y aprobaciones

Utilice esta tabla para el control de revisiones y aprobaciones del procedimiento.

Versión	Fecha dd/mm/aaaa	Autor	Estado	Revisado por	Aprobado por	Descripción modificación
1.0	16/10/2014	Miguel Camilo Páez Pirazan				Inicio de documento procedimiento propuesto de pentesting

ANEXO B Ejecución del procedimiento

1 Propósito del documento

El propósito de este documento es mostrar la ejecución del procedimiento propuesto.

2 Alcance

El procedimiento que se describe en este documento fue realizado basándonos en la metodología OWASP, con la cual se busca minimizar la superficie de ataque que pueda tener un aplicativo web desarrollado por la fábricas de software, los controles que fueron seleccionados se tomaron a partir de la validación del top 10 de las amenazas más comunes publicado por el OWASP, cubriendo del siguiente modo dichas amenazas:

1. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Inyección de código malicioso, redireccionamientos inválido a otros sitios de un portal, objetos referenciados directamente), mediante los controles (OWASP-IG-003 “Identificar puntos de entrada”, OWASP-DV-001 ”Reflected XSS” y OWASP-DV-005 ”SQL injection”) descritos en este procedimiento.
2. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Configuración incorrecta), mediante los controles (OWASP-IG-004 “Testing for web applciation fingerprint”) descritos en este procedimiento.
3. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Información sensitiva expuesta), mediante los controles (OWASP-CM-005 “Testing for file extensions handling” y OWASP-IG-001”Spiders and crawlers”) descritos en este procedimiento.
4. El procedimiento cubrió la validación de las siguientes vulnerabilidades (Usar componentes con fallas conocidas), mediante los controles (OWASP-IG-004 “Testing for web application fingerprint” y OWASP-CM-001 “SSL/TLS Testing (SSL Version)”) descritos en este procedimiento.
5. Adicionalmente en el procedimiento se ejecuta un análisis de vulnerabilidades el cual será usado como refuerzo a las validaciones anteriores y el cual da chance para realizar validaciones manuales.

Para realizar la ejecución del procedimiento el usuario debe basarse en el archivo que contiene los pasos a ejecutar (Anexo: Procedimiento_Pentest_v1.0.docx), por ende este documento solamente contendrá el objetivo de cada punto y la evidencia de ejecución.

3 Definiciones

A continuación se presentan algunos términos utilizados en este documento que hacen relación al proceso de pruebas.

Términos	Definición
Pentest	Es como comúnmente se denomina a los (Test de penetración) y son en conjunto la forma de denominar a una serie de técnicas utilizadas para evaluar la seguridad de redes, sistemas de computación y aplicaciones involucradas en los mismos
Entregable	Resultado tangible (documento, formato, software, etc.) de una tarea completada.
Tester	Analista de pruebas que ejecuta las pruebas. *Ingeniero Miguel Camilo Páez Pirazan*

4 HERRAMIENTAS A USAR

- **Kali de Linux:** Es un Proyecto open source que es mantenido y fue fundado por Offensive and security, un proveedor de clase mundial de entrenamiento en seguridad informática y servicios de test de penetración. Kali de linux es una distribución GNU/Debian diseñada principalmente para auditoria y seguridad informática en general.

- **Web Scarab:** Webscarab es un marco de trabajo para analizar aplicaciones web que se comunica usando los protocolos HTTP y HTTPS. Está escrito en Java, por lo que es portable a muchas plataformas. WebScarab tiene muchos modos de operación, implementados por varios plugins. Su uso más común es operar WebScarab 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.

- **Nmap:** Es un programa de código abierto, que se usa para escanear puertos, servicios y descubrir servidores en una red informática.

- **Zenmap:** Es el GUI oficial de seguridad de Nmap, se trata de una plataforma multi (Linux, Windos, Mac Os entre otros), el cual tiene como objetivo tener a disposición las mismas funcionalidades de nmap, pero de un modo gráfico y más sencillo de utilizar para principiantes y expertos en nmap.

- **Openssl:** Es una herramienta de criptografía que implementa SSL y TLS protocolos de red y los estándares requeridos por estos. Este software es desarrollado y mantenido por una comunidad abierta de voluntario con gran conocimiento en seguridad de la información.

- **Accunetix:** Accunetix es una herramienta que puede ser capaz de escanear sitios web en busca de fallos de seguridad que puedan poner en peligro la integridad de un sitio web publicado en internet.

- **SQLMAP:** Sqlmap es una herramienta de pentesting open source que automatiza la el proceso de detección y explotación de fallas de sql injection.

- **Nikto:** Nikto es un web server scanner que realiza test contra aplicaciones web, usando múltiples validaciones varios items, como lo son archivos y directorios potencialmente peligrosos, validación de actualizaciones entre otros.

5 PROCEDIMIENTO

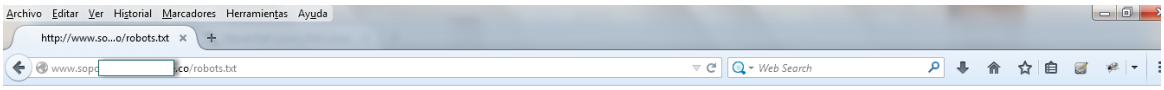
A continuación se mostrará la ejecución del procedimiento en donde se debe dejar plasmada la evidencia de ejecución por medio de toma de pantallas.

a. Validar existencia de archivo robots.txt (Control OWASP-IG-001)

Objetivo: Realizar la verificación de la existencia del archivo “Robots.txt”, con el cual se podrá validar si existen directorios dentro del portal que no se quieran ser indexados por los buscadores como google, bing etc.

Ejecución: Para la ejecución de este punto se realizó la verificación de la existencia del archivo mediante la consulta del archivo por medio de la URL (ver Figura 1).

Figura 1 Resultado Robots.txt




```
#
# robots.txt
#
# This file is to prevent the crawling and indexing of certain parts
# of your site by web crawlers and spiders run by sites like Yahoo!
# and Google. By telling these "robots" where not to go on your site,
# you save bandwidth and server resources.
#
# This file will be ignored unless it is at the root of your host:
# Used:    http://example.com/robots.txt
# Ignored: http://example.com/site/robots.txt
#
# For more information about the robots.txt standard, see:
# http://www.robotstxt.org/wc/robots.html
#
# For syntax checking, see:
# http://www.sxw.org.uk/computing/robots/check.html

User-agent: *
Crawl-delay: 10
# Directories
Disallow: /includes/
Disallow: /misc/
Disallow: /modules/
Disallow: /profiles/
Disallow: /scripts/
Disallow: /themes/
# Files
Disallow: /CHANGELOG.txt
Disallow: /cron.php
Disallow: /INSTALL.mysql.txt
Disallow: /INSTALL.pgsql.txt
Disallow: /INSTALL.sqlite.txt
Disallow: /install.php
Disallow: /INSTALL.txt
Disallow: /LICENSE.txt
Disallow: /MAINTAINERS.txt
Disallow: /update.php
Disallow: /UPGRADE.txt
Disallow: /xmlrpc.php
# Paths (clean URLs)
Disallow: /admin/
Disallow: /comment/reply/
Disallow: /filter/tips/
Disallow: /node/add/
Disallow: /search/
Disallow: /user/register/
Disallow: /user/password/
Disallow: /user/login/
Disallow: /user/logout/
# Paths (no clean URLs)
Disallow: /?q=admin/
Disallow: /?q=comment/reply/
Disallow: /?q=filter/tips/
Disallow: /?q=node/add/
Disallow: /?q=search/
Disallow: /?q=user/password/
Disallow: /?q=user/register/
Disallow: /?q=user/login/
Disallow: /?q=user/logout/
```

Fuente: De Autores

Como se puede validar dentro del portal se encuentra el archivo robots.txt, en el cual se ven listados los archivos y folders a los cuales los buscadores no tendrán acceso. Este proceso de validación tardó 10 minutos.

b. Validar putos de entrada del portal (Control OWASP-IG-003)

Objetivo: Verificar dentro del portal a analizar cuáles podrían ser posibles puntos de ataque, como lo pueden ser formularios dentro del portal (ingreso, autenticación, registro etc.), tratando de validar así una superficie de ataque con posibles campos no validados correctamente, para realizar esta validación se tendrán en cuenta los métodos GET y POST que usa la aplicación en sus operaciones.
Ejecución: A continuación veremos que sucede cuando tratamos de acceder al portal por medio del formulario de login posterior a la configuración de webscarab.

Webscarab empieza a generar unas alertas, mediante las cuales veremos los métodos GET y POST que son usados y lanzados por la aplicación, y que pueden

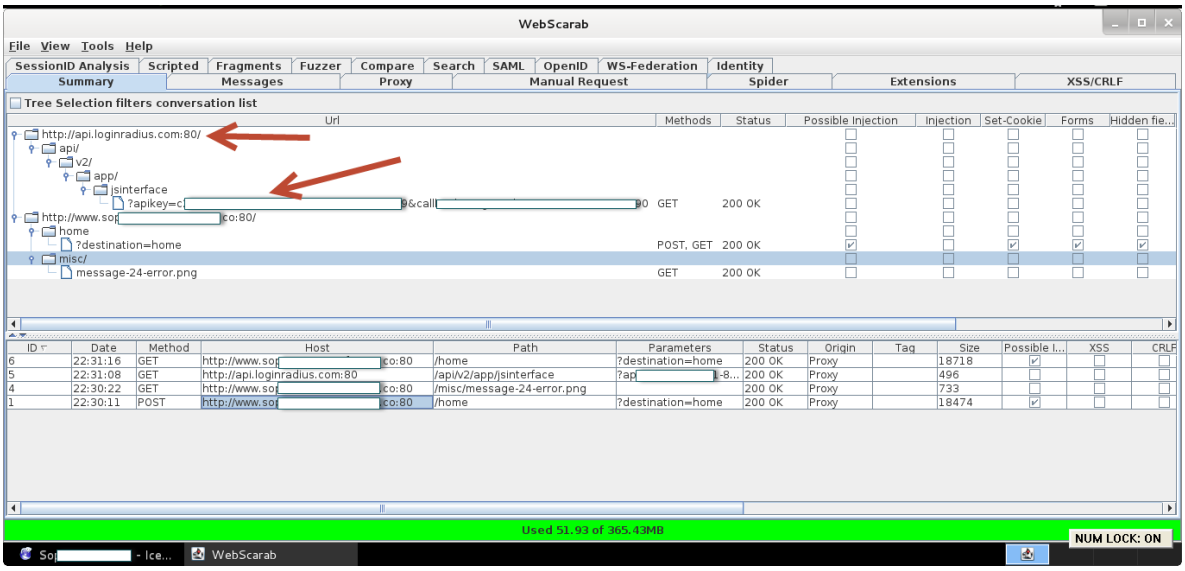
ser cambiados antes de llegar al servidor, adicionalmente nos informa que tipo de método se acabo de capturar (Ver Figura 2).

Figura 2 Alerta de intercepción WebScarab vs Formulario Login

Fuente: De Autores

Como nos encontramos en la etapa inicial de recolección de información, no modificaremos ningún valor, solamente validaremos los métodos capturados por el proxy webscarab cuando tratamos de autenticarnos y validando así que pueden existir puntos de entrada (Ver Figura 3).

Figura 3 Resultado de intercepción completa formulario Login



Fuente: De Autores

Como se puede apreciar en la Figura 3 se ve toda la traza de comunicación con el servidor, los métodos usados y las variables que fueron usadas al enviar la petición de autenticación, adicionalmente se puede apreciar que se encontró una comunicación con un sitio llamado www.api.loginradius.com, este sitio es utilizado para realizar la autenticación de los usuario sobre el portal con IDs ya existentes, como por ejemplo facebook, flirck etc.[27], al cual se le envía una llave de identificación del sitio. Este proceso de validación tardó para este portal validando los diferentes formularios que este contiene 1 hora, sin embargo cabe aclarar que este tiempo puede ser diferente dependiendo de la cantidad de páginas, formularios y estructura que pueda tener un portal.

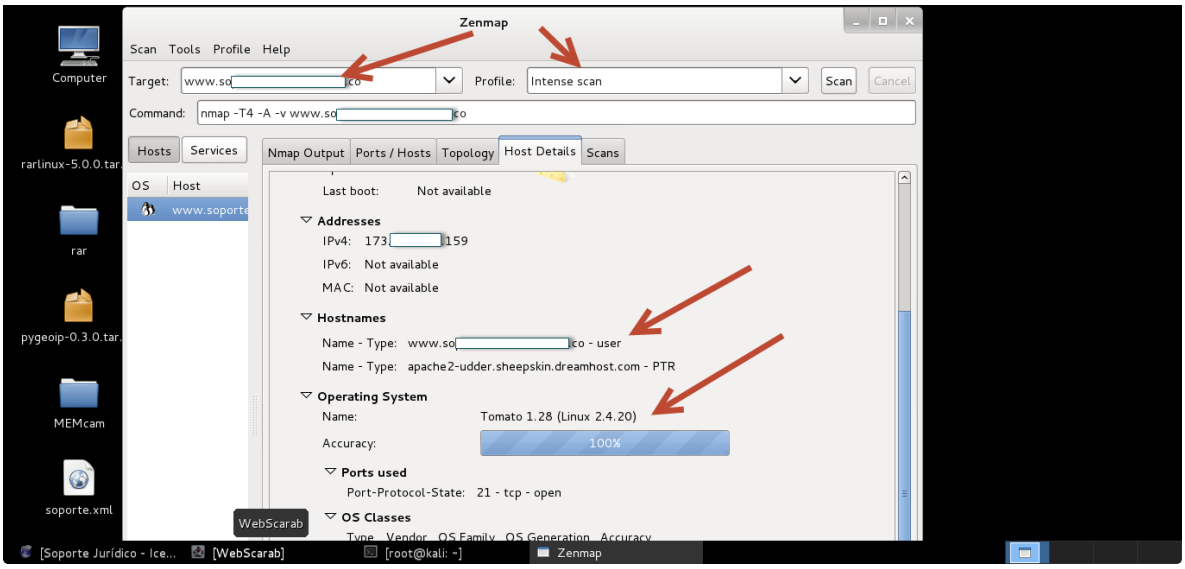
c. Validar fingerprint del servidor de aplicaciones (Control OWASP-IG-004)

Objetivo: Verificar primero que todo si la configuración del servidor permite al tester obtener la versión del servidor de aplicaciones en el cual el aplicativo está instalado y a partir de esto validar si la versión del servidor que se está usando está actualizado y tiene alguna vulnerabilidad conocida.

Ejecución: En está validación se verá a continuación los resultados arrojados por el aplicativo (Ver Figura 4).

Figura 4 Resultado ejecución zenmap

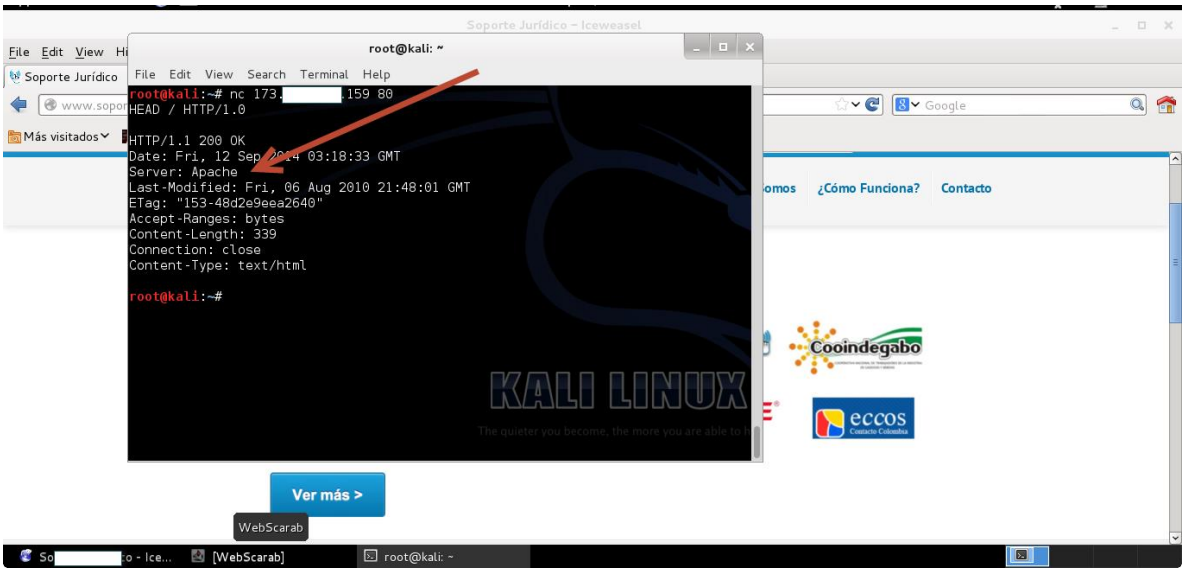
[27] Wikipedia: LoginRadius [en línea]. < http://en.wikipedia.org/wiki/Loginradius>. [Citado en 2015]



Fuente: De Autores

Ahora para saber la versión del servidor ejecutaremos un comando de netcat, con el cual podremos saber que versión de servidor de aplicaciones tiene la aplicación. En una terminal de Linux, ejecute el siguiente comando (netcat nc xxx.xxx.xxx.xxx puerto) y después de haber digitado esto, en la parte inferior ejecutar el siguiente comando (HEAD / HTTP/1.0), tal como se ve en la siguiente imagen en la cual se puede ver que el servidor es Apache (Ver Figura 5). El tiempo de ejecución que tomó esta validación fue de 40 minutos, 30 minutos realizando la validación con el software zenmap y 10 minutos la validación con netcat.

Figura 5 Resultado de ejecución comando netcat



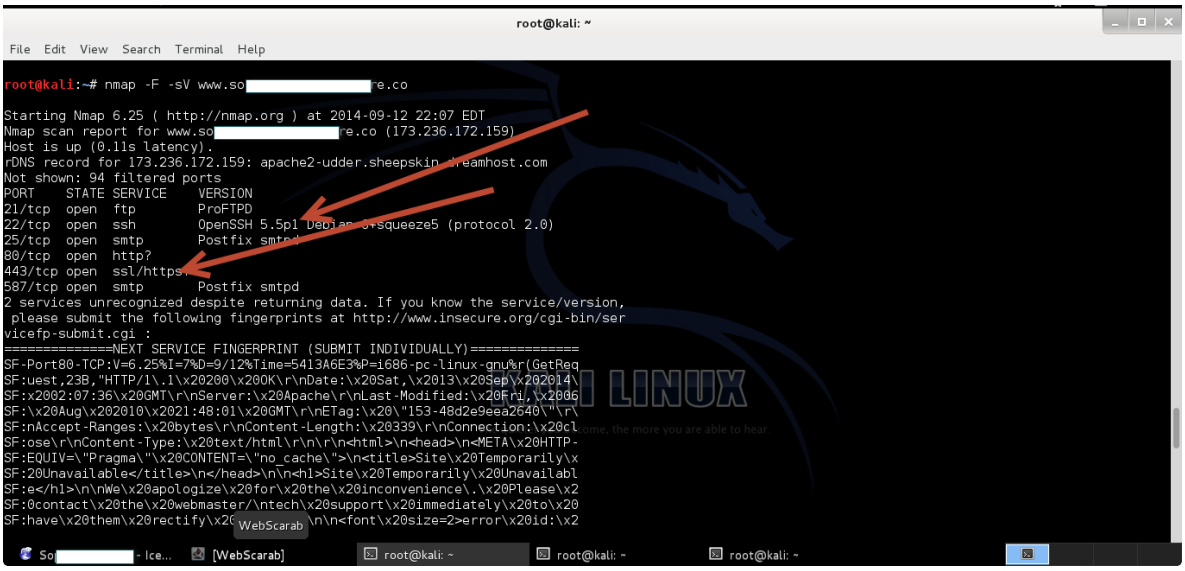
Fuente: De Autores

d. Validar métodos de encripción (Control OWASP-CM-001)

Objetivo: Verificar si en el servidor existen métodos de encripción seguros, y adicional a esto verificar si estos están actualizados o no.

Ejecución: En ésta validación se puede validar los métodos de encripción usados por el servidor mediante la ejecución de los comandos descritos en el procedimiento (Ver Figura 6).

Figura 6 Resultado Nmap

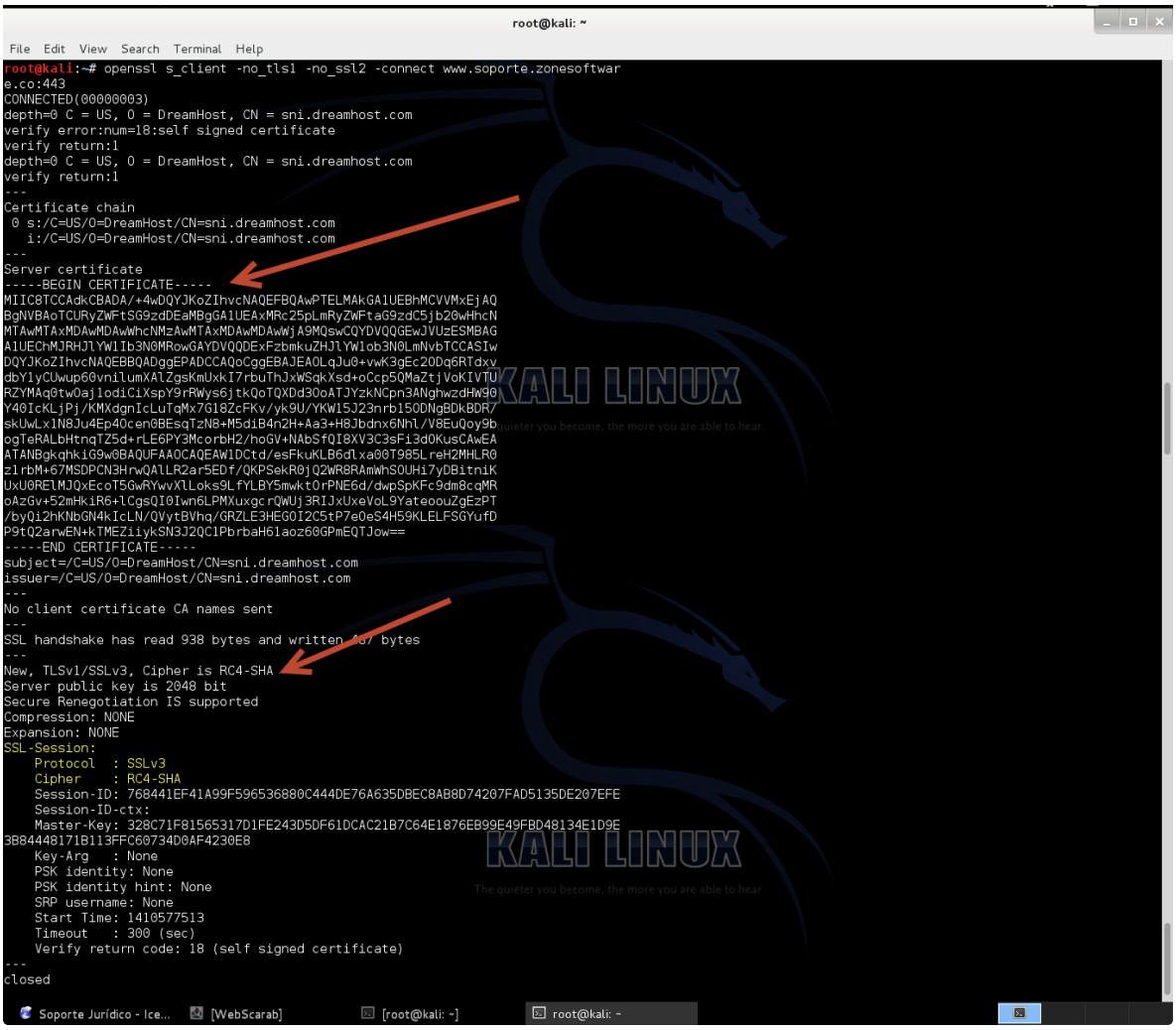


Fuente: De Autores

Como podemos validar el comando ejecutado nos muestra como resultado que si hay un servicio SSL corriendo en dos posibles puertos, como lo vemos en la imagen, en el puerto 22 y el puerto 443, por ende con Openssl validaremos ambos puertos para ver en cual está corriendo realmente el servicio.

Ahora y a partir de la información obtenida en la ejecución anterior, validaremos la versión de SSL y TLS usados, mediante el uso de la herramienta Openssl. Ejecutaremos el siguiente comando en una terminal Linux (openssl s_client -no_tls1 -no_ssl2 -connect www.host.co:443), si el resultado es exitoso el sistema nos mostrará la información del servidor como, versión de ssl y tls, certificado del servidor y algoritmo de encripción usado. El comando (s_client, es usado para crear un cliente SSL / TLS genérico para conectarnos a un host remoto), los comando (-no_tls1 y –no_ssl2, son usados para conectarnos al host remoto omitiendo los protocolos nombrados, en este caso omitiría la conexión con sslv2 y tlsv1 y trataría de conectar con el servidor por medio de sslv3), y por último el comando (-connect www.host.co:443, es usado para especificar a qué servidor remoto nos queremos conectar y por medio de que puerto).

Figura 7 Resultado de ejecución comando Openssl



Fuente: De Autores

Como podemos validar se verificó que efectivamente el servidor contiene la versión TLSv1/SSLv3 (Ver Figura 7), que a este momento son las versiones que se están usando. La validación de este proceso tardó 30 minutos.

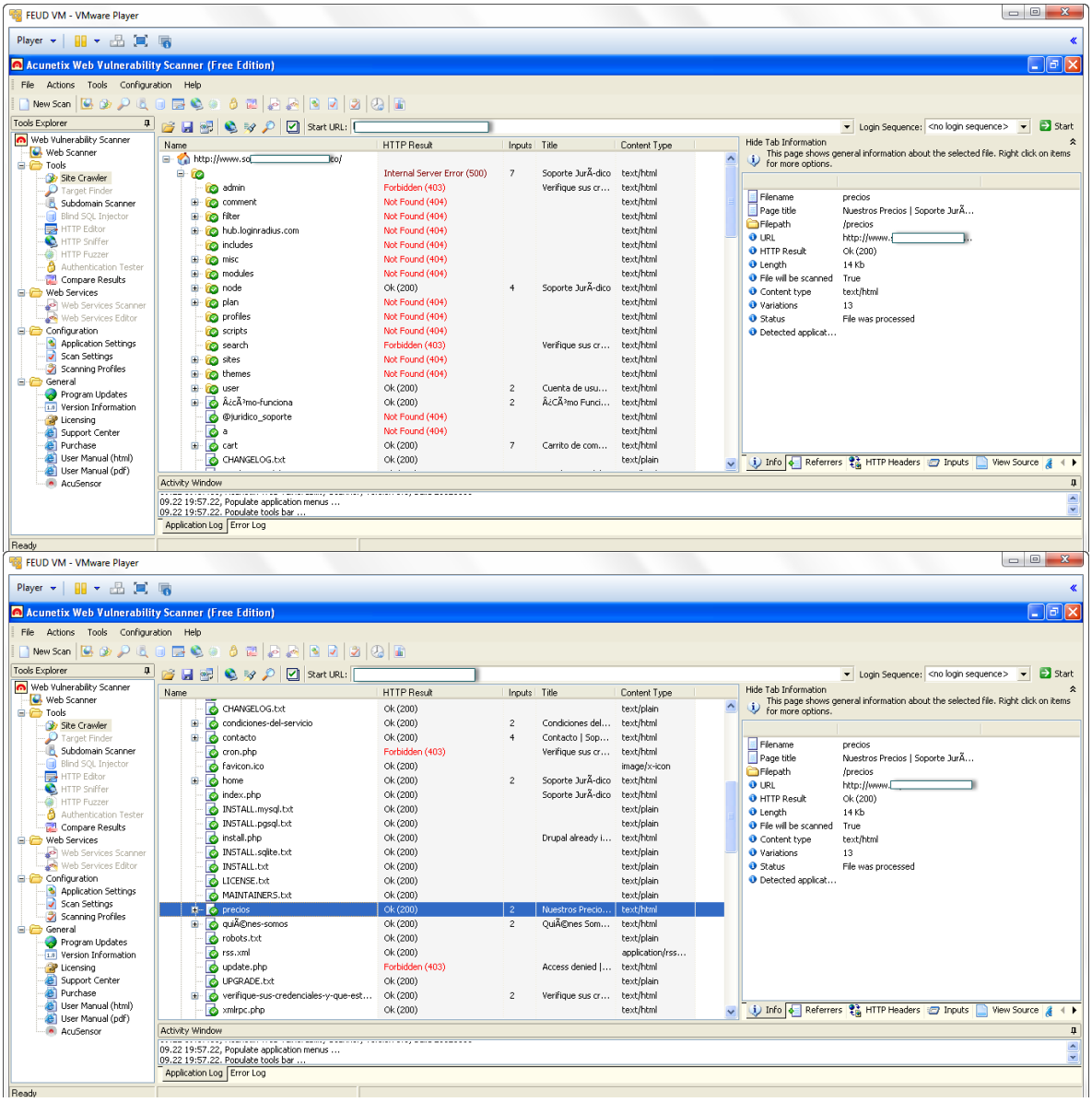
e. Validar configuración y manejo de archivos (Control OWASP-CM-005)

Objetivo: Verificar si el portal está manejando bien los archivos, evitando que esté expuesto al usuario información sensible o archivos de configuración.

Ejecución: Después de haber realizado la ejecución del procedimiento teniendo en cuenta las configuraciones previas que se deben realizar para ejecutar el web

scanner, el sistema nos arrojará como resultado la estructura del portal que este scanner haya encontrado al igual que archivos (Ver Figura 8).

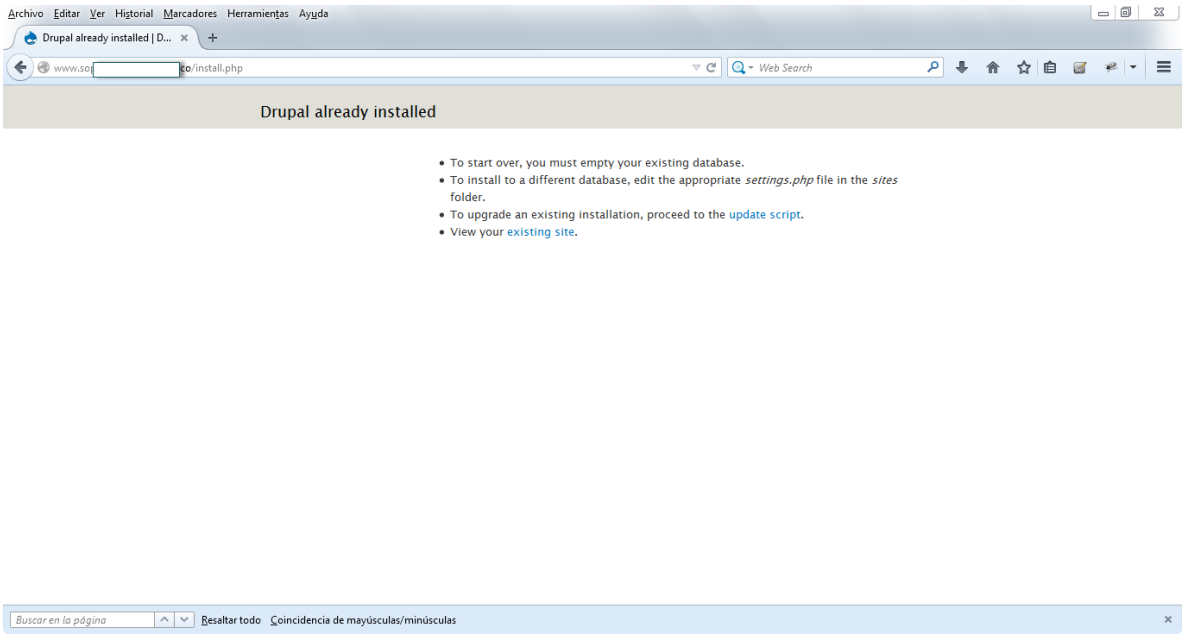
Figura 8 Resultado web crawler accunetix



Fuente: De Autores

Del scanner anterior se tomo uno de los archivos listados por el scanner validando así la existencia del archivo “install.php” como se ve en la imagen a continuación (Ver Figura 9).

Figura 9 Revisión de archivo install.php



Fuente: De Autores

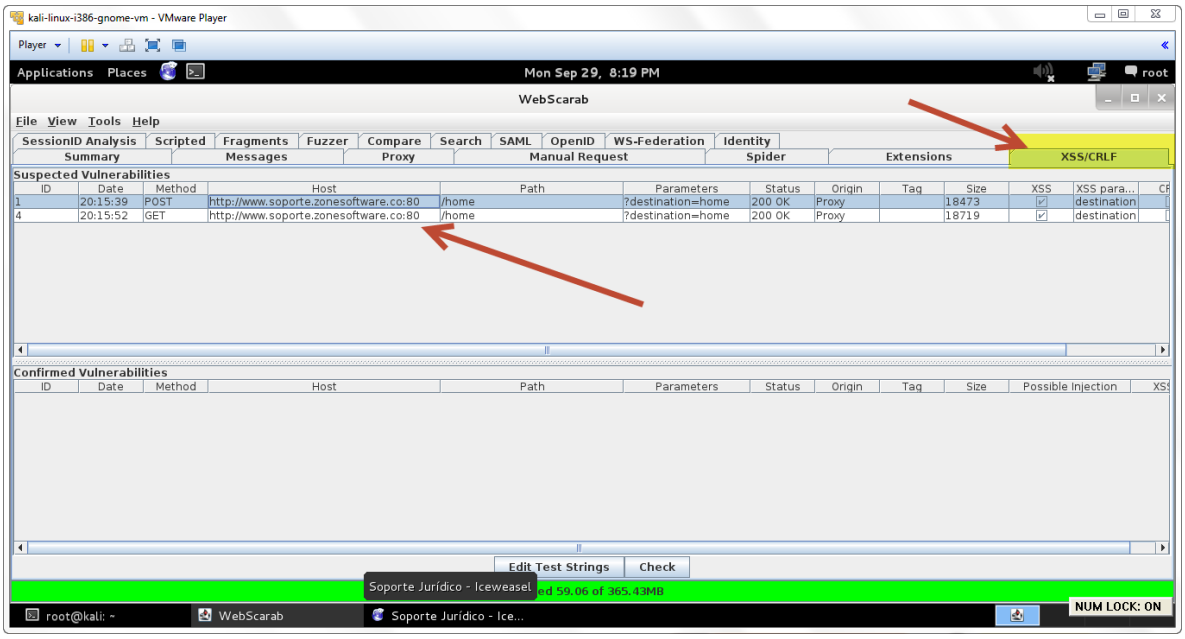
La ejecución de este proceso tardó aproximadamente 3 horas, este proceso no solo depende del equipo en el cual se esté ejecutando el aplicativo sino del servidor se esté ejecutando la prueba, por ejemplo en la ejecución de este proceso al enviarse múltiples solicitudes simultaneas el servidor presentaba fallas, por lo cual se tuvo que pasar el proceso varias veces hasta que el servidor respondía nuevamente.

f. Validar existencia de inyección cross site scriptng y CRLF (Control OWASP-DV-001)

Objetivo: Verificar si de alguno de los puntos de entrada validados en el punto (6.2 Control OWASP-IG-003) se puede sacar ventaja utilizando la vulnerabilidad de XSS ó CRLF. Este tipo de prueba consiste en tratar de inyectar código javascript o html, por ejemplo en formularios, variables pasadas por la URL etc.

Ejecución: Teniendo en cuenta el proceso explicado en el procedimiento a continuación se mostrarán los resultados generados a partir de las validaciones y configuraciones realizadas (Ver Figura 10).

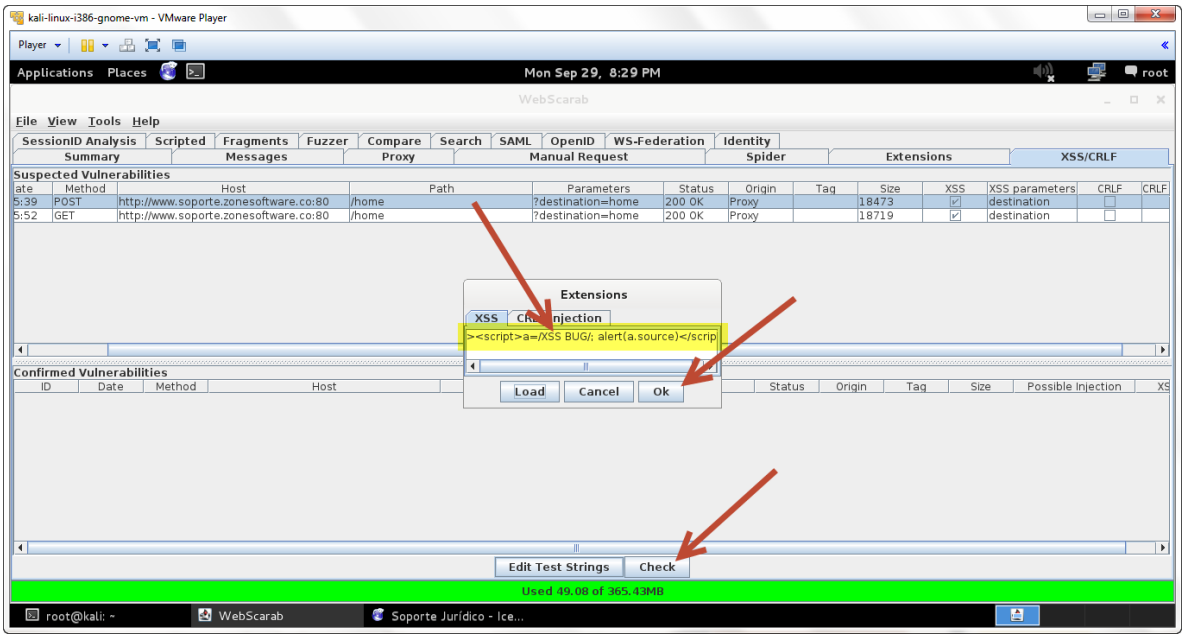
Figura 10 Resultado Webscarab para cross site scripting



Fuente: De Autores

Después de haber listado las url navegadas, webscarab le permite probar de una manera manual las url que posiblemente pueden tener ataque la vulnerabilidad de cross site scripting, seleccionando de la lista la URL a probar, de click en el botón “Edit Test Settings” e ingresando allí el valor con el cual desea probar la posible vulnerabilidad, de click en ok y después para probar la vulnerabilidad de click en check (Ver Figura 11), si después de dar click en check la URL aparece en la grilla inferior, quiere decir que la vulnerabilidad existe, de lo contrario es porque no existe dicha vulnerabilidad en la URL seleccionada.

Figura 11 Configurar webscarab para probar XSS en URL



Fuente: De Autores

Como se pudo validar dentro del portal se verificaron las URL listadas por webscarab, dando como resultado que ninguna de las URL tenía este tipo de vulnerabilidad. El proceso de esta validación tardó 1 hora realizando la validación de las URL.

g. Validar existencia de SQL inyección (Control OWASP-DV-005)

Objetivo: Validar si existe dentro del portal verificado alguna vulnerabilidad asociada a Sql Injection, que permita a un atacante obtener información que está contenida en la base de datos y sea de uso exclusivo del aplicativo.

Ejecución: En el portal que se está validando para la empresa se pudo corroborar que no posee la vulnerabilidad de sql injection, pues se probaron varias de las URL que podrían contener este tipo de vulnerabilidad y no se encontró.

h. Ejecutar análisis de vulnerabilidades Web

Objetivo: Verificar con un scanner de vulnerabilidades web si existen dentro del portal vulnerabilidades adicionales que no hayan sido verificadas.

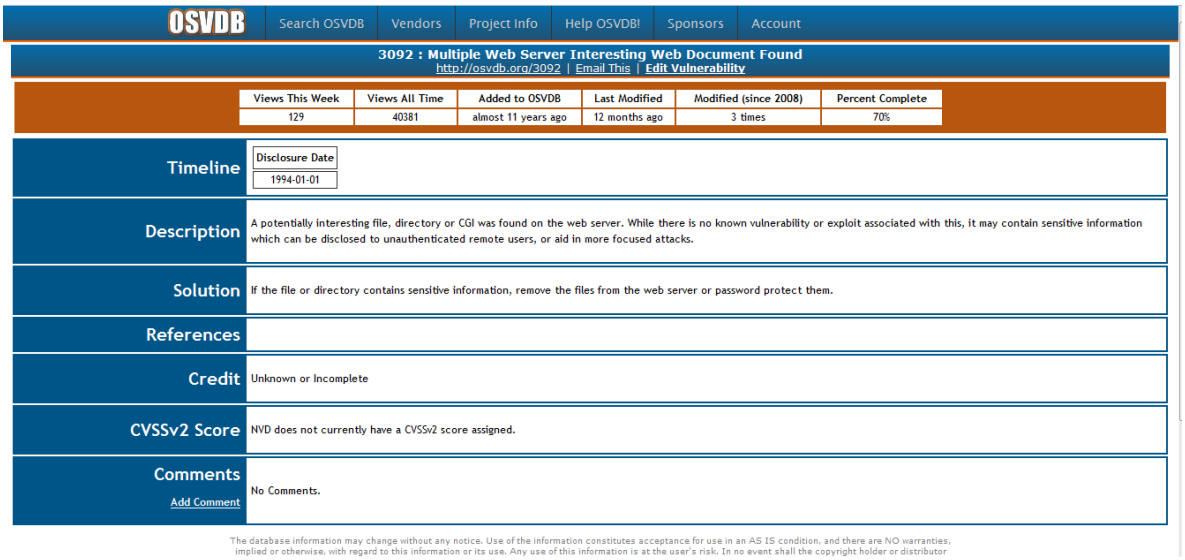
Ejecución: A continuación se presentará el log generado por nikto dada la ejecución del escaneo web en búsqueda de vulnerabilidades que no hayamos detectado. La ejecución de este proceso tardo 3 horas, esto sucedió porque el servidor sobre el cual estaba instalado el aplicativo web no soportaba multiples conecciones

simultaneas, por ende era necesario pausar el proceso y reiniciarlo cuando el servidor volvía a responder.

```
root@kali:~# nikto -host www.soporte.zonesoftware.co
- Nikto v2.1.4
---------------------------------------------------------------------------
+ Target IP:          173.236.172.159
+ Target Hostname:    www.soporte.zonesoftware.co
+ Target Port:        80
+ Start Time:         2014-09-25 19:22:51
---------------------------------------------------------------------------
+ Server: Apache
+ robots.txt contains 36 entries which should be manually viewed.
+ OSVDB-3092: /web.config: ASP config file is accessible.
+ OSVDB-3092: /cart/: This might be interesting...
+ OSVDB-3092: /error_log: This might be interesting...
+ OSVDB-3092: /home/: This might be interesting...
+ OSVDB-3092: /user/: This might be interesting...
+ OSVDB-3092: /UPGRADE.txt: Default file found.
+ OSVDB-3092: /install.php: Drupal install.php file found.
+ OSVDB-3092: /install.php: install.php file found.
+ OSVDB-3092: /LICENSE.txt: License file found may identify site software.
+ OSVDB-3092: /xmlrpc.php: xmlrpc.php was found.
+ OSVDB-3233: /INSTALL.mysql.txt: Drupal installation file found.
+ OSVDB-3233: /INSTALL.pgsql.txt: Drupal installation file found.
+ 6448 items checked: 174 error(s) and 21 item(s) reported on remote host
+ End Time:          2014-09-26 21:42:35 (94784 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

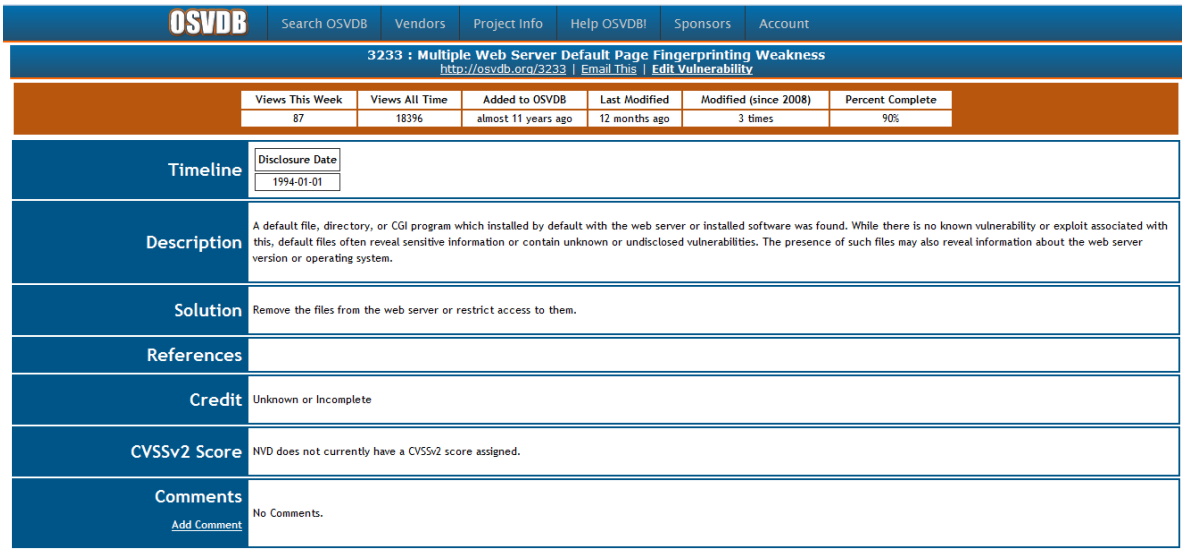
A continuación se muestran el resumen de las vulnerabilidades encontradas por nikto (Ver Figura 12 y Figura 13).

Figura 12 Informe de vulnerabilidad OSDVBD 3092



Fuente: http://osvdb.org/3092

Figura 13 Informe de vulnerabilidad OSDVBD 3233



Fuente: http://osvdb.org/3233

6 Recomendaciones

- Se valido dentro del portal que la principal vulnerabilidad del aplicativo está asociada a cuanta información tiene este expuesto, pues como se pudo validar hay muchos archivos expuestos que pueden ser consultados y que pueden contener información sensible, por ende se recomienda remover los

archivos que no sean usados y restringir el acceso a directorios que no deban ser navegados por los usuarios finales.

- Se pudo validar que en algunas ejecuciones donde se enviaban múltiples solicitudes al portal este presentaba fallos debido a la cantidad de solicitudes enviadas por el usuario, por ende se sugiere que para este tipo de pruebas el servidor sea un poco más robusto para que las ejecuciones puedan fluir mejor.
- También se recomienda validar que los problemas que se tuvieron con el portal referente a las caídas por peticiones múltiples no se deba a problemas con el aplicativo sino con la configuración del servidor, puesto que si esto sucede en producción es muy posible que se puedan realizar ataques de denegación de servicio de este modo.

7 Conclusiones

- La ejecución de las pruebas tardaron en total 9 horas con 10 minutos, teniendo en cuenta los imprevistos explicados en cada uno de las ejecuciones.
- El tamaño del sitio juega un papel importante dentro de la ejecución de este tipo de pruebas, pues no es lo mismo realizar el escaneo a un sitio que contenga 100 páginas diferentes a uno que contenga solo 10.
- El servidor en donde el aplicativo se encuentre instalado es muy importante, pues como nos pudimos dar cuenta en este aplicativo el servido no era muy robusto, lo cual hacía que se cayera el aplicativo por momentos en los escaneos de vulnerabilidades.

8 Revisiones y aprobaciones

Utilice esta tabla para el control de revisiones y aprobaciones del procedimiento.

Versión	Fecha dd/mm/aaaa	Autor	Estado	Revisado por	Aprobado por	Descripción modificación
1.0	16/10/2014	Miguel Camilo Páez Pirazan				Inicio de documento procedimiento propuesto de pentesting